2-2011

A Teleological Approach to Robot Programming by Demonstration

John Douglas Sweeney
University of Massachusetts Amherst

A TELEOLOGICAL APPROACH TO ROBOT PROGRAMMING BY DEMONSTRATION

A Dissertation Presented

by

JOHN D. SWEENEY

Submitted to the Graduate School of the

University of Massachusetts Amherst in partial fulfillment

of the requirements for the degree of

DOCTOR OF PHILOSOPHY

February 2011

Computer Science

A TELEOLOGICAL APPROACH TO ROBOT PROGRAMMING BY DEMONSTRATION

A Dissertation Presented

by

JOHN D. SWEENEY

Approved as to style and content by:

Roderic A. Grupen, Chair

Oliver Brock, Member

Andrew H. Fagg, Member

Rachel Keen, Member

Andrew G. Barto, Department Chair
Computer Science

To my parents.

ACKNOWLEDGMENTS

The help of many people was instrumental in creating this dissertation. Foremost is my advisor, Rod Grupen. I am very grateful for all of the guidance and support he has given me throughout my graduate studies. I have learned so much from his insight and will always appreciate his consideration and candor. Rod has been a wonderful mentor, showing me how to do meaningful research in robotics, peel back the frontier of knowledge, and make robots do cool things. His warm and friendly manner set the tone for the lab, and helped make it such an enjoyable place to work and learn. For all of this I am truly thankful.

I would also like to thank the members of my committee for their help and advice throughout this journey. I would like to thank Oliver Brock for always keeping his office door open to me; I thoroughly enjoyed our many conversations. Oliver is always eager to share his enthusiasm and deep knowledge of robotics and he introduced me to many new areas of the field, for which I am grateful. I am also indebted to Andy Fagg for introducing me to Dexter, and mentoring my first work on the robot. It was from our initial collaboration that the work in dissertation was born. I greatly respect and admire Andy's breadth of knowledge and technical abilities, both in and out of robotics. Thanks to Rachel Keen for her insightful perspective on the intersection between robotics and developmental psychology. Bouncing ideas off her and the other members of her lab was always a fruitful exercise.

I also owe a huge debt of gratitude to all the members of the Laboratory for Perceptual Robotics, past and present, that I had the pleasure of working with, and that helped shape the course of my studies: TJ Brunette, Patrick Deegan, Stephen Hart, Emily Horrell, Chao Ou, Rob Platt, Chandu Ravela, Shiraj Sen, and Bryan Thibodeau. A special thanks to Michael Rosenstein for his advice and for sharing the helpful teleoperation software libraries he developed for Dexter.

I am truly fortunate to have been able to pursue a degree in robotics at this university. The Computer Science department is filled with interesting people doing great research. My friends in the department have made this time in Amherst so enjoyable and rewarding, for which I am immensely thankful.

I would like to thank my globe-trotting brother, Tim, for his support and enthusiasm for my work throughout this process. His music provided the soundtrack for many late nights working in the lab. I would also like to thank my parents for their love and support, and for providing me with the opportunity to pursue my interests, academic and otherwise. For this I am eternally grateful.

I would especially like to thank my wife Casey, for her never-ending support of me in this endeavor. Without her love, encouragement and understanding, I would never have been able to complete this work. All of my love to you.

ABSTRACT

A TELEOLOGICAL APPROACH TO ROBOT PROGRAMMING BY DEMONSTRATION

FEBRUARY 2011

JOHN D. SWEENEY

B.Sc., CARNEGIE MELLON UNIVERSITY

M.Sc., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Roderic A. Grupen

This dissertation presents an approach to robot programming by demonstration based on two key concepts: demonstrator intent is the most meaningful signal that the robot can observe, and the robot should have a basic level of behavioral competency from which to interpret observed actions. Intent is a teleological, robust teaching signal invariant to many common sources of noise in training. The robot can use the knowledge encapsulated in sensorimotor schemas to interpret the demonstration.

Furthermore, knowledge gained in prior demonstrations can be applied to future sessions.

I argue that programming by demonstration be organized into declarative and procedural components. The declarative component represents a reusable outline of underlying behavior that can be applied to many different contexts. The procedural component represents the dynamic portion of the task that is based on features observed at run time. I describe how statistical models, and Bayesian methods in particular, can be used to model these components. These models have many features that are beneficial for learning in this domain, such as tolerance for uncertainty, and the ability to incorporate prior knowledge into inferences. I demonstrate this architecture through experiments on a bimanual humanoid robot using tasks from the pick and place domain.

Additionally, I develop and experimentally validate a model for generating grasp candidates using visual features that is learned from demonstration data. This model is especially useful in the context of pick and place tasks.

TABLE OF CONTENTS

Page

ACKNOWLEDGMENTS . . . . . **v**

ABSTRACT . . . . . **vii**

LIST OF TABLES . . . . . **xv**

LIST OF FIGURES . . . . . **xvi**

CHAPTER

1. INTRODUCTION **1**

1.1 Research Approach 3

1.2 Contributions 6

2. RELATED WORK **8**

2.1 Methods for Acquiring Training Data 9

2.2 Abstraction in PbD Systems 11

2.2.1 Procedural Abstraction 12

2.2.2 Declarative Abstraction 17

2.3 The Teleological Approach 20

2.3.1 Goals 22

2.3.2 Means 24

2.3.3 Constraints 24

2.3.4 Action Selection 25

2.4 Biological Motivation 26

3. THE CONTROL BASIS AND SENSORIMOTOR SCHEMAS 29

3.1 The Control Basis 30

3.1.1 Artificial Potential Approaches in Robotics 35

3.1.2 Multi-Objective Control 36

3.1.3 Primitive Controllers 37

3.1.3.1 A Controller for Reaching 37

3.1.3.2 A Controller for Manipulability 39

3.1.3.3 A Controller for Grasping 40

3.2 Turning Controllers into Behavior 40

3.2.1 Discrete State Dynamics 41

3.2.2 Sensorimotor Behaviors 43

3.2.2.1 Search-Track 44

3.2.3 Hierarchical Structure 46

3.2.3.1 Reach-Grasp 46

3.3 Representing Complex Behavior with Schemas 48

3.3.1 Declarative and Procedural Decomposition of Controllers 48

3.3.2 Sensorimotor Schemas 50

3.3.3 Pick-And-Place 51

3.4 Discussion 53

4. A DECLARATIVE REPRESENTATION FOR PROGRAMMING BY DEMONSTRATION 54

4.1 Declarative Representation of Schemas 55

4.1.1 Hidden Markov Models 56

4.2 Interpreting Demonstration 61

4.3 Applying Knowledge from Demonstration 66

4.3.1 Experimental Apparatus 66

4.3.2 Experiment: A sorting task 68

4.3.3 Discussion 73

5. PROCEDURAL INFORMATION FROM DEMONSTRATION 74

5.1 Generative Models for Procedural Policies 76

5.1.1 A Generative Model for Pick and Place Proceduralization 80
5.1.2 Experiments with a Sorting Task Using Pick and Place 84
5.1.2.1 Sorting Based on Color 86
5.1.2.2 Sorting Based on Size 91
5.1.3 Prediction of Task Relevance 94
5.2 Incremental Learning 100
5.2.1 Multi-color Sorting Experiment 101
5.3 A General Framework for Procedural Policies from Schemas 107
5.3.1 Modeling Feedback Element Assignments 107
5.3.2 Modeling Effector Resource Assignments 109
5.3.3 Mapping Features to Feedback Elements 111
5.3.4 Creating a Feature Model 113
5.4 Discussion 116
6. INTERACTING WITH OBJECTS 117
6.1 Related Work 122
6.2 Representing Grasp Prototypes in the Model 123
6.2.1 Visual Appearance 123
6.2.2 Hand Position and Orientation 124

Position 124
Orientation 125
6.3 The Generative Model 126
6.4 Parameter Estimation in the Model 129
6.4.1 Generating Pre-Grasps for New Objects 132
6.5 Experimental Results 133
6.5.1 The Naïve Model 136
6.6 Discussion 141
7. CONCLUSIONS AND FUTURE WORK 143
7.1 Contributions 144
7.2 Looking Forward 147
APPENDICES
A. INTRODUCTION TO BAYESIAN MODELS 150
A.1 Nonparametric/Semiparametric Bayesian Models 151
A.1.1 Maximum Likelihood Estimates 152
A.1.2 Bayesian Modeling 154
A.1.3 Mixture Models 161

A.1.4 Dirichlet Process 164

A.1.5 Inference using Monte Carlo Methods 166

B. OBJECTS USED IN EXPERIMENTS 169

B.1 Objects Used in Experiments 170

BIBLIOGRAPHY 178

LIST OF TABLES

Table **Page**

5.1 **Red objects.** This table shows how each object can be used to recognize other objects using the discrete color features. A • indicates a recognized object; blank indicates unrecognized. 90

6.1 Each `x` denotes a grasp on the object in the row that was used by the grasp prototype denoted in the column. The number of `x`'s in row i is the number of grasp prototypes used by object i. The number of `x`'s in column j is the number of objects that use that grasp prototype j. 140

LIST OF FIGURES

Figure **Page**

1.1 An overview of the programming by demonstration architecture explored in this dissertation. 5

3.1 A closed-loop controller. The function G transforms the error signal e into a control signal u for the robot. The function H measures the output of the system and produces a sensor signal used to compute the error signal. 30

3.2 A schematic of the control basis. The controller C is composed from the basis sets of sensor, effector, and signal transforms. At the top level, the gradient of the artificial potential ϕ creates a reference signal $\boldsymbol{u}_\tau$ for effector resource τ. The underlying closed-loop controller actuates τ, which generates a sensor stream σ, that defines the current position in the potential field. 34

3.3 The dynamics of two controller activations. The grey area, $\dot{\phi} \leq \epsilon$, represents the quiescent state for the controller; when the dynamic state enters this region, the controller is considered to be converged. Controller A has reached a quiescent state where $\phi = a$. Controller B has not yet reached quiescence, but has already reached a lower absolute level of the artificial potential than A. 42

3.4 The quad state graph is a state machine describing the run-time dynamics of a control basis behavior. The starting state $*$ represents an undefined state in the underlying controller. The state $\varnothing$ denotes an undefined reference. The states 1 and 0 represent whether the controller has converged or not, respectively. 43

3.5 The Search-Track behavior. If a tracking reference is not found when the behavior begins, it activates the Search controller. This controller runs until it has located a tracking reference, at which point the Track controller is activated to foveate on the target. 45

3.6 The Search-Track behavior using a multi-objective controller. In this instance, the Search and Track controllers are co-activated, and they are prevented from destructively interfering by using null space projection. 46

3.7 The Reach-Grasp behavior can initiate the Search-Track behavior if a reaching target is undefined. After Search-Track has found a target, the behavior executes the multi-objective Reach controller subject to a Grasp controller. 47

3.8 The Pick-And-Place Schema. The Reach-Grasp behavior is used to initially grasp the object, and then the multi-objective Reach subject to Place controller is executed. 52

4.1 The hidden Markov model. The grey nodes are observed variables. At each time step, an observation is generated conditional on the latent variable. The next state variable is chosen from a distribution conditioned on the value of the current state. 58

4.2 The organization of monitors into an observation vector. In this figure, there are three monitor sets for controllers i, j, and k. Each monitor set consists of a group of monitors with different controller references. The state variable $\boldsymbol{x}_t$ is composed of the concatenation of the convergence status of each monitor set. 64

4.3 Each of Dexter's arms has 7 DOF and is equipped with a three-fingered hand with four total degrees of freedom. The stereo head has four degrees of freedom. 67

4.4 The teleoperator's viewpoint. The icon in the upper right denotes whether the teleoperator is actively controlling the robot. The graph in the upper-left denotes the convergence status of the active schema. The red circle denotes the current state, while the dark red circles are previously visited states. 68

4.5 A sorting sequence, clockwise from top left. The sorting task is to place orange objects in the right bin, and blue objects in the other. The original demonstration used Dexter's right arm and bins in different locations. In this execution, Dexter autonomously uses its left arm to sort the ball into the left bin. 70

4.6 A sorting sequence, clockwise from top left. In this execution, the robot must sort the orange object into the right bin. This requires transferring the object between the workspaces of the two arms. This illustrates the contingency event of arm to arm transfer in the Pick-And-Place schema. 72

5.1 The simpler pick and place model used in this chapter. This model describes a single instantiation of pick and place. The variable q represents the latent place category for the pick and place instance with a hyperparameter β. The observed variable r_{P} is the feedback element for the Place controller. This variable is conditioned on K distinct places x. Each of the K categories has a unique distribution θ with hyperparameter γ; sampling from one of these distributions produces the n observed features in a training example. 79

5.2 This plot shows the success rate for sorting objects across 10 trials for each number of example objects. 88

5.3 This plots the maximum p-value (out of 10 trials) per number of examples used. This p-value measures the significance of the sortable object classification. 89

5.4 This figure shows the classification label assigned to each object presentation, based on the size of the object segment. 91

5.5 The posterior distribution over size after different numbers of training examples per category. This figure represents the same model, but each line represents the posterior updated to include a given number of training examples of each category. The distribution is bimodal, with the peak on the left accounting for Small objects, while the peak on the right describes Large objects. 92

5.6 The posterior classification between Small and Large for every object instance, with varying number of training examples. 93

5.7 The Normal-Inverse-Gamma prior distribution; note the logarithmic scale of the Variance axis. The mean is modeled by the Normal component, and the variance by the Inverse-Gamma distribution. 95

5.8 This shows the ROC curve for classifying whether an object should be sorted or not, with various number of examples. 97

5.9 The actual task classification: each object instance is marked as sortable (□) or ignorable (∘). In this task, only Medium objects were ignored. 98

5.10 The posterior classification of task relevance. Each graph shows the learned model after a specific number of training examples, denoted by black ∗. The blue dashed line represents the learned MAP decision boundary. The magenta dashed line represents the maximum likelihood decision boundary. 99

5.11 This shows the evolution of the probability mass functions; the x-axis corresponds to the set of 64 discrete colors. The top row shows the Red/Black category, and the bottom shows Blue/White. Each column, from left to right, shows the effect of an additional example provided by the demonstrator. 102

5.12 Colors generated from each pmf, as new examples are provided. Each slice represents a collection of 36 samples from the posterior distribution. Each sample is represented as a block in the slice, with the order of the blocks within the slice determined by frequency. Each sample represents one of 64 different HSV regions, and the color of the region is shown by pixels taken uniformly from the region. The slices from left to right show the effect of an additional training example. The top row corresponds to the Red/Black task, and the bottom to Blue/White. As examples are added, each slice shows the posterior distribution becomes closer to the true Red/Black or Blue/White distribution. 104

5.13 This figure shows how the model changes incrementally. The left column shows an image of the object presented for training, order from top to bottom in the order in which they were presented. The second column shows the object after color quantization into one of 64 different color values. The remaining images are constructed from the quantized image by filling in pixels based on the likelihood of the segments color according to the learned distribution. For example, the top-most row shows the likelihood of the blue bottle according to the posterior after each additional object is presented. 106

5.14 This shows the different types of feedback elements. Reference r_f is a task-independent reference, typically for force-based controllers. Reference r_i is a task dependent reference, and r_j is a reference dependent on r_i. 110

5.15 A generic feature model for a schema with two references. 115

5.16 This shows the different possible relationships between q and the feature types. The lack of an arrow between nodes indicates no statistical relationship exists between the variables. 116

6.1 This figure shows how the mallet can be segmented into multiple segments by construction, where each segment can be used to generate grasp positions independently. 117

6.2 This shows different grasp locations generated by the model in Section 6.5. Each frame shows a pre-grasp associated with a distinct grasp prototype. 119

6.3 The graphical model described in Section 6.3. Circles indicate random variables, all unshaded variables are latent, shaded variables are observed. A rectangle around nodes represents replication, with the number of replications written in the bottom right corner. The edges between nodes indicates a conditional probability relationship described in the text. There are M objects and A learned visual grasp prototypes, where object m has N_m observations. θ represents the parameters of a multinomial distribution over the set of visual grasp prototypes. z is an indicator variable for one of the A prototypes. The observed variables b, x, and w are the object's visual appearance feature, and the relative hand/object position and orientation, respectively. The bottom row shows the latent distribution parameters for each of the A visual grasp prototypes: ψ and u are define the inverse-Wishart distribution for the visual feature, μ and Σ are the mean and covariance of the normal distribution describing the Cartesian position of the relative hand/object pose, and ϕ is the parameter of the multinomial distribution over hand orientation. 127

6.4 This picture shows the objects in the training set. The red oval corresponds to the covariance matrix that was used as a visual feature for grasps with the object. 134

6.5 This picture shows the objects as they were presented for generating grasps.The red oval corresponds to the covariance matrix that was computed from the average second moments of the segmented blob in the left and right cameras. 135

6.6 This shows different pre-grasps generated by the model in Section 6.5. The top three pre-grasps were generated by the same grasp prototype, while the bottom two pre-grasps came from two different grasp prototypes. 136

6.7 This graph shows the result of using the trained grasp model on a set of test objects. Each bar measures the number of successful grasps for the labeled object. The blue bars are for the naïve model, and the red for the visual grasp prototype model. 137

A.1 The model on the left shows the Gaussian mixture model $\mathcal{M}$ in expanded form. The graph on the right uses the shorthand notation of "plates" to denote replication across variables. The α and λ variables are hyperparameters, denoted using rounded-edge boxes. 164

B.1 There are 24 object presentations classified as Red. 171

B.2 There are 18 object presentations classified as Blue. 172

B.3 There are 17 object presentations classified as White. 173

B.4 There are 12 object presentations classified as Black. 174

B.5 There are 38 object presentations classified as Small. 175

B.6 There are 39 object presentations classified as Medium. 176

B.7 There are 31 object presentations classified as Large. 177

CHAPTER 1

INTRODUCTION

As humanoid robot technology matures, these platforms will become a common presence in research, industrial, and home settings. We expect these robots to accomplish complicated tasks that can be performed in multiple ways under myriad conditions. Moreover, they must be able to handle uncertainty in the environment as a result of incomplete sensor information and a consequence of unstructured, open domains. These robots, with many degrees of freedom (DOF) and multiple sources of sensory input, have the benefit of enormous flexibility, but at the cost of complexity.

A requirement for a useful, general purpose humanoid robot is that it can be programmed in the field by non-experts. Consider a humanoid robot designed for use in a home setting. Although the robot may leave the factory with a number of programmable abilities, the specifics of how the robot should use its skills will depend on the home in which it is placed. Therefore the robot will require *in situ* programming, through a combination of autonomous learning and instruction, to successfully complete tasks. This process should be simple, intuitive, and require a minimal amount of communication. This dissertation proposes an intermediate approach where instruction via teleoperation is complemented by learning techniques that internalize the information conveyed by instruction.

Programming can be considered a search problem through the robot's configuration space, the size of which grows exponentially with the number of DOF. For humanoids, this size renders brute-force approaches infeasible. Traditional programming approaches solve this problem by requiring the operator to explicitly specify the features, conditions, and goals of the task. This can be challenging, even for a robotics expert, and is likely impossible for non-experts. Instead, the robot should be taught to perform tasks in a way that is natural for humans: instruction through demonstration.

Programming by demonstration (PbD) addresses the huge search space associated with a complex robot by using the demonstration as a way of narrowing that space into areas that are most promising. The demonstration provides a jumping-off point for autonomously finding improved solutions in novel contexts. The challenge in PbD is how to extract a meaningful signal from the demonstration into a form that the robot can use.

If the robot has a basic sensorimotor competence, these skills can provide perceptual and motor abstractions that further reduce the search space and equip the robot with a sensorimotor vocabulary from which to construct an interpretation of demonstration. Demonstrations can be explained as rearrangements or elaborations of these underlying, simple behaviors. In this thesis, I argue that a competent, low-level relationship between the robot and the environment is a prerequisite for understanding demonstration.

1.1 Research Approach

Any successful PbD system must address two fundamental questions: What about the demonstration should the robot imitate? and How should the robot imitate the demonstrator? These two questions are answered by the declarative and procedural aspects of the programming system, respectively.

The declarative dimension of a PbD system—what to imitate—is a representation of the actions and goals that can be achieved in a domain, and represented by a task model that allows the robot to make novel inferences. The declarative aspect describes the complexity of the task model that can be learned from demonstration, and consequently, the objective function of the system [20].

The procedural aspects of a system—how to imitate—range along a spectrum of state and action space abstractions. At one end are the most explicit representations, and at the other are approaches that use high levels of abstraction. The level of abstraction in a PbD system is proportional to the functional "power" of the approach. That is, the lower the level of the state/action space, the less generalizable the learned skill, and the less universal are the concepts conveyed from teacher to student. A further benefit of abstraction is an increase in teaching efficiency. The instructor can communicate salient features of the demonstration in a more abstract space, which requires less bandwidth.

In this thesis, I propose two key concepts that address the declarative and procedural aspects of the PbD problem. The first is that the most meaningful signal that the robot can extract from demonstration is the intent of the teacher. Intent is more robust than, say, trajectory. However, unlike trajectory, intent is a latent variable that must be inferred by the observer. Humans' ability to infer the intentions of

others is an active area of research among psychologists [130, 58, 8, 59], and I have been motivated by the so-called teleological hypothesis [65, 66] as a road map for how a robot might accomplish these inferences.

The second concept is that the observing robot should be supplied with a fundamental level of behavioral competence. Sensorimotor schemas—motor programs that encapsulate a limited range of behavior—are used to endow the robot with a basic set of abilities. In this procedural approach, the robot uses schemas to parse the demonstration event stream into a simpler, more abstract training signal. After training, the robot uses all of the background knowledge and preferences it has acquired throughout its lifetime to execute the task—a key distinction in this work.

By combining the intent of the demonstrator with knowledge in the form of sensorimotor schemas, a unique result of this approach is that the robot can execute the task in ways that were never demonstrated, using features that are invariant to many changes in the work space and configuration space of the demonstrator. Figure 1.1 shows a graphical overview of the architecture I present in this work.

To demonstrate the framework presented in this thesis, I focus on the problem domain of object manipulation in the context of pick and place tasks. Pick and place encompasses a wide variety of useful behavior that can be found in many types of robot platforms. Many types of object manipulation tasks can be reframed as pick and place tasks; for instance, assembly and sorting tasks. The main experimental platform used in this work is Dexter, the UMass bimanual humanoid. The usual demonstration method is for a human operator to control Dexter remotely through some task; this is known as teleoperation.

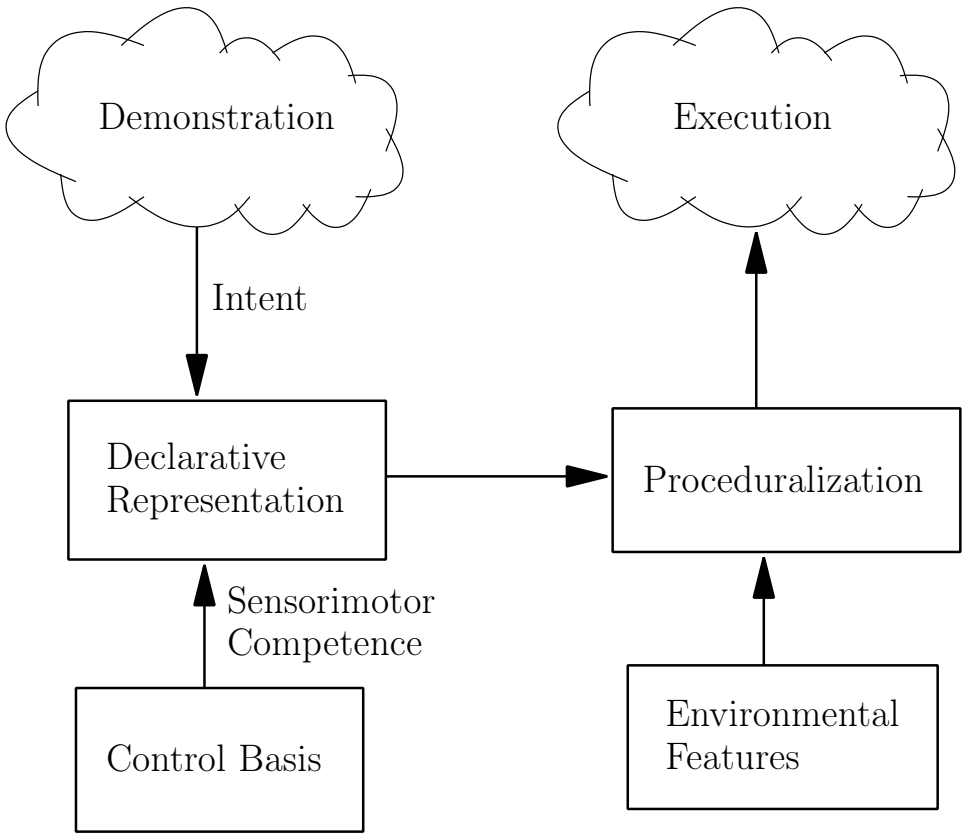

Figure 1.1. An overview of the programming by demonstration architecture explored in this dissertation.

1.2 Contributions

This thesis advances the state of the art with two main contributions.

1. I develop a schema-based, computational architecture for skill transfer from demonstration. The declarative structure allows the robot to deal with contingencies when executing tasks learned from demonstration. The architecture is experimentally validated with a series of pick and place experiments using teleoperation.

2. I propose a computational model of visual grasp prototypes that is used to learn the relative hand and object pose prior to performing grasp from demonstration. I use a statistical approach to representing procedural knowledge provided by demonstration.

With regard to the first contribution, one of the crucial distinctions in this work is the assumption that the robot interprets a demonstration in terms of an existing behavioral vocabulary. The control basis architecture is the behavioral framework I use for this vocabulary [31]. As described in Chapter 3, closed-loop controllers are assembled by associating sensor and effector resources with navigation functions that represent objectives or "intentions." These controllers form the "basis" of behavior for the robot, and are the building blocks used to develop sensorimotor schemas. These schemas, described in Chapter 3, are used by the robot to interpret and execute programs given by the demonstrator. I focus on the pick and place schema in this work because of its general utility in object manipulation tasks in humanoid domains.

In Chapter 4, I describe a method for inferring declarative representations of demonstrated tasks. This model attempts to find the existing behavioral schema that best

matches the observed behavior. It uses convergence events of closed-loop controllers to identify the intention of the demonstrator.

In Chapter 5, I describe a method for learning procedural models from demonstration. The goal is for the robot to make generalizations about the task from the demonstration; this represents a transfer of knowledge from the teacher to the robot. Using the statistical models and manipulation routines developed in earlier chapters, I explore how the robot can successfully apply knowledge gained from demonstration to novel task scenarios.

My second main contribution is a model for learning grasp prototypes from demonstration, described in Chapter 6. This model uses nonparametric Bayesian techniques to infer posterior distributions over pre-grasps—the position and orientation of the hand prior to grasping. The robot uses visual features of the object to generate pre-grasps that result in improved grasping performance, even with naïve grasp controllers.

Throughout this work, statistical models are used to deal with uncertainty in the teaching signal and provide a way to generalize knowledge presented in a demonstration. I use Bayesian models because of their flexibility in density representation, and for their ability to integrate prior knowledge into the model.

CHAPTER 2

RELATED WORK

The functionality of any robot can be described in the most basic terms of states it can observe and actions it can produce in response. Let $\mathcal{S}$ be the set of states that represents all the available information about the environment. There may exist a subset of $\mathcal{S}$ that is *hidden* state and represents information about the environment that is unavailable to the robot's direct perceptual processes. Let $\mathcal{F}$ be the set of *features* that the robot uses to represent the perceptual state. There is a function $f : \mathcal{S} \rightarrow \mathcal{F}$ that maps states to features. The set $\mathcal{A}$ represents the actions available to the robot. At the most basic level, the functionality of a robot can be described in terms of a *policy*: a mapping $\Pi : \mathcal{S} \rightarrow \mathcal{A}$, from states to actions.

For humanoid robots in open environments with many degrees of freedom, the state and action spaces are intractably large, and the discernible features and actions grow with experience. The challenge of PbD is to develop the appropriate state and action space representations that allow the robot to learn and execute policies based on training data. This data is composed of observations of the demonstrator executing a target policy, and every PbD system must use some method to collect this data.

2.1 Methods for Acquiring Training Data

An instructor can employ a wide variety of techniques for demonstrating training knowledge, each providing different amounts and quality of information to the robot. On one end of the spectrum are methods where the robot experiences the task first-hand. This type of presentation involves the teacher performing an example solution directly, using the robot's body, along the way indicating important events. This explicit teaching input can be achieved by teleoperation: a process that translates motions of the operator into motions of the manipulator [154]. Typically the operator also has feedback from the workspace produced by the robot's sensors. These techniques allow the robot to observe how the recommended strategy appears to its sensors and how its mechanical abilities can be coordinated to produce results. This approach has the benefit that the expert performs the task directly, and the instruction is tacit—there are no explicit statements about the target task or workspace [101].

One degree removed from teleoperation is the use of sensor suits or motion capture systems to record the movements of the teacher performing the task [85, 86, 109, 151]. In this case, the robot has access to the history of joint angles and velocities of the demonstrator, whose kinematics, dynamics, feedback, and motor abilities may differ significantly from the robot. Finding a suitable mapping of the demonstrator's morphology onto the observer is known as the "correspondence problem" [122].

The most "remote" method of capturing kinematic and dynamic information is passive observation of an uninstrumented demonstrator. In this case, the morphological correspondence problem is harder, as the robot must infer the joint angle and velocity history of the teacher from the sensor stream. When a non-direct demonstration method is used, observation is more complicated because the robot must solve the

correspondence problem, and it must infer how feedback events inform control decisions. As the observation becomes more remote—and the teaching more implicit—it is more likely that the state and action spaces between the instructor and the robot will diverge. This presents the additional problem of developing a translation between the state and action spaces of the teacher to those available to the robot. Moreover, morphological correspondence is challenging to construct and can be misleading, because morphological equivalence does not necessarily imply functional equivalence. For example, although a human may use their finger to push a button, it may be better for a humanoid to use a different part of its arm instead, due to differences in kinematic and dynamic capabilities of the robot.

One important feature common to all these approaches is that the robot does not have access to the knowledge the teacher uses to arrive at the instructive demonstration, but the teacher believes that the robot can distinguish and appreciate these distinctions. While the robot receives kinematic and dynamic information in different forms with varying degrees of noise, in all cases, the intended goal of the teacher is initially hidden state that must be inferred to transfer the strategy effectively to the pupil. It is the goal of the instructor to make this conspicuous, and of the pupil to associate discernible events with appropriate actions. A good teacher will attempt to convey the target concept in many different ways in order to emphasize the common attributes across different task contexts. This encourages the student to focus on the common declarative features of the skill while making the irrelevant procedural details less salient. For example, a teacher may demonstrate how to operate a switch using a hand, foot, or elbow, in order to emphasize the more important declarative feature of actuating the switch rather than the method used to actuate it.

The above-mentioned techniques lie along a spectrum from explicit to implicit conveyance of information about new tasks. However, it is possible to conceive of a situation where the approaches are mixed. For example, the main form of demonstration could consist of passive observation, but with the demonstrator executing more complex subtasks using a direct approach such as teleoperation. This hybrid technique is commonly used when teaching people how to hit a golf ball, for example. The instructor might first perform a few swings for the student, and then physically move the student's arms through the motions during their first swing.

Whether the instruction is explicit or implicit, the robot observes motor trajectories derived from the demonstrator's policy. These observations are a high bandwidth, noisy stream of artifacts of intentional action. The robot must filter this based on its own integrated behavior, to extract the intentions conveyed within. I contend that the robot should infer a policy at a high level of abstraction in state and action space: at the level of intention. Thus, the value of the demonstration is to convey high level intentions rather than low level motor artifacts. The challenge in designing a PbD system is to develop a representation that facilitates simultaneous teaching and learning at the right level of abstraction.

2.2 Abstraction in PbD Systems

In this section, I organize existing work using a dichotomy of procedural and declarative focus. The procedural abstraction—how to imitate—describes the representational approach used by the robot to execute tasks it has observed. These approaches can vary from explicit joint space representations that use little to no abstraction, to

higher level state and action space representations that use multiple levels of abstraction.

The declarative representation—what to imitate—is a model of the task being demonstrated. This can also range from explicit objectives, for example, "Move the end effector to location $\mathbf{x}$," to high level abstractions that require inferring the action to perform. As task goals become more abstract, it is important for the robot to extract intention from the demonstration rather than explicit artifacts of the actions displayed by a third-party demonstrator.

2.2.1 Procedural Abstraction

In most PbD sessions, the demonstration will likely be suboptimal. The demonstrated trajectory may not be smooth, and it is likely that certain parts of the task may fail and be repeated during the course of the demonstration. For example, the teleoperator may grasp an object and try to place it in a certain location, but drop the object *en route*. Procedural abstraction provides a way to mitigate the effects of suboptimal demonstration.

At the lowest level of abstraction are approaches that represent the demonstration as a full trajectory. In this approach, the robot performs an optimized version of the demonstrated trajectory that minimizes discrepancies with a target trajectory. Yeasin and Chaudhuri developed a system to extract smoothed trajectories by clustering sets of 3D trajectories describing the motion of the demonstrator's hand captured by a vision system [159]. The resulting trajectories are presented to the user for selection and execution. Ude *et al.* describe a system for building a kinematic model of the demonstrator from motion captured marker data [151]. Using this model

and a wavelet basis representation, they infer smoothed joint trajectories from the demonstration. However, trajectories only refer to the task indirectly, and very similar trajectories can be elicited for very different tasks.

Grudic and Lawrence detail a nonparametric approach to representing the sensor stream experienced by an expert controlling the robot through a task [71]. Their approach creates a dimensionally-reduced, compressed approximation of the mapping between the sensor stream observed by the demonstrator and the corresponding commands for each motor in a tracked mobile robot. However, it can be difficult to interpret the learned parameters of the model and determine what exactly was learned.

This very low level instruction space is not limited solely to position-control trajectories. Asada and Izumi describe a system where the robot learns a set of hybrid force/position trajectories for a given task instance [6]. The resulting program is a very simple repetition of the trajectory performed by the demonstrator. Delson and West also detail a method for learning a force/position controller in a constrained environment [41]. Multiple demonstrations provide clues to the robot about the environmental constraints and workpiece misalignment. In such *ad hoc* applications, the generalizability of the approach is limited, as the robot must be retaught when the environment changes. In addition, planning and control in the high dimensional configuration space characteristic of typical humanoid robots is computationally challenging due to the curse of dimensionality.

Some researchers choose to perform explicit trajectory reproduction in the operational space [98, 120] of the robot instead of the higher dimensional configuration space. While operational space represents a higher level of abstraction, many PbD

systems still exhibit limited state/action abstraction by learning end effector trajectories. Aleotti and Caselli use a clustering algorithm and a polynomial spline-based method to create a smoothed end-effector trajectory from a set of exemplars [1]. Asada *et al.* use a visual servoing technique based on the epipolar constraint to imitate end effector trajectories of a demonstrator robot [7]. Yokokohji *et al.* use an Extended Kalman Filter (EKF) to track the demonstrator's hand and to produce end effector trajectories using vision [160].

Billiard and Matarić describe a biologically inspired system that aims to reproduce the characteristics of human arm motions using a hierarchical, connectionist framework [16]. Although they use a hierarchical framework for interpreting the demonstration, their state space remains at the level of joint trajectories.

Other research uses demonstration data as a "bootstrap" for optimizing a controller. Atkeson and Schaal describe a system for learning to imitate a human performing a pendulum swing-up task [9, 10]. They have one trial of human demonstrator data, and use a model of the task to form an optimal control problem. To learn how to swing the pendulum up, they use a feedforward controller that uses the distance from the demonstrated trajectory along with the distance of the pendulum from the goal configuration as optimization criteria. The goal of such "direct policy learners" is to mimic both the trajectory of the device and the trajectory shown by the demonstrator [141]. This method of learning from demonstration requires a deep knowledge of the task *a priori*, and is task-specific. Moreover, this task is one in which the final geometry *is* the goal, and therefore implicitly, and completely, captures the task.

In the domain of assembly tasks, there has been work to describe the demonstration in terms of the different contact states of the robot and objects involved in the task [81,

26, 27]. In many cases, this research has focused on a specific application and its imitative requirements, and the approach does not generalize to other domains, or even other tasks.

Hayes and Demiris [79] present a system for imitative learning in mobile robots that associates local sensor perceptions to the perceived actions of the teacher. The result of learning is an implicit configuration space trajectory represented by a set of associative perception-action rules. This approach is more robust than an explicit configuration space trajectory, but overall it does not scale to more complex platforms.

Dixon *et al.* describe a system for predictively generating waypoints for a robot manipulator programming task [45]. Their system learns to predict the next waypoint given the recent history of waypoints chosen by the user. Waypoints are represented as points in workspace using a distance-based clustering method, where the compactness of the resulting model is set as a parameter.

Approaches that focus on trajectory reproduction are inherently limited by the "narrowness" of the learned policy. The trajectories are not robust to changes in workspace and task constraints. However, for some specific tasks, repeatability and precision can be achieved at the expense of generalizability. My goal is to contribute technology for addressing the generalizability of policies gleaned from PbD methods.

Sensorimotor Primitives

An important aspect of abstraction concerns the ability to segment trajectories into smaller, repeatable parts defined by invariant features of the end state. Behavioral

primitives are sensorimotor programs that provide a natural way to break up the complex actions of a humanoid into more manageable elements. Each primitive represents a (possibly parameterizable) policy defined on a subset of the state/action space. There is biological support for the concept: evidence suggests that vertebrates generate motions through linear combinations of motor primitives [117].

There have been many different approaches for both representing and acquiring primitives; many researchers have developed primitives to serve the needs of a particular application. Some authors have used smoothed segments of the demonstrator's configuration space trajectory to represent motor primitives [44, 94]. Tominaga and Ikeuchi define primitives in terms of contact relationships in the configuration space of the manipulator [150]. One of the downsides of the configuration space approach is that it is a highly nonlinear space, and the curse of dimensionality can make configuration space approaches intractable for humanoids with many DOF.

Other research has proposed representing primitives as spatiotemporal paths on a lower dimensional manifold that describes the original demonstrated trajectory compactly [109, 110, 87, 88]. Dimensionality reduction abstracts the demonstrated trajectory by projecting it onto a subspace, where each dimension in the new space can represent coordinated patterns of motion in the original degrees of freedom. Voyles *et al.* construct a set of linear motor primitives using principal component analysis to create a dimensionally reduced motor action space from demonstration data [157, 156]. Each primitive represents the sensor and actuator signal projected onto the subspace defined by a set of principal components.

Schaal *et al.* discuss the use of *dynamic movement primitives* for programming by demonstration [142]. They have defined two classes of primitives, one for point at-

tractive motions, and another for limit cycle movements. Each primitive is a set of autonomous, nonlinear dynamical systems for a collection of DOF of the robot that defines a control policy. Ijspeert *et al.* implemented a programming by demonstration system for a humanoid based on point attractive [86] and limit cycle [85] versions of these primitives. They used the primitives to classify movements of the demonstrator by comparing the velocity outputs of different (prior-learned) control policies with the velocities of the demonstrator. Primitives may also be defined *a priori* by a domain expert. For example, to enable a humanoid to learn to play air hockey from demonstration, Bentivegna *et al.* define an *ad hoc* set of primitives specific to the domain [13].

In this thesis, I also take a primitive-based approach, with closed-loop controllers serving as the behavioral basis. The difference in my approach is that motor primitives are closed-loop controllers native to the control basis and not learned from demonstration, discussed in Chapter 3. Complex behavior is constructed by combining these controllers in specific ways based on the information conveyed in a demonstration. Moreover, controllers provide a form goal abstraction, in that they are capable of generating many different trajectories but maintain the invariant of a reproducible end state.

2.2.2 Declarative Abstraction

The declarative abstraction of a PbD system describes the task model used to represent the actions and abstract objectives of a domain. In the context of humanoid PbD, many papers have focused on the declarative aspect of the problem by exploring

how the task space can be represented; declarative representations are also a subject of traditional artificial intelligence research [4, 153, 60].

The motor primitive approach provides a convenient way to decompose the task space: each individual subtask corresponds to a parameterization of a primitive. These subtasks can serve as the building blocks for a traditional production-rule planning system [53]. Kuniyoshi *et al.* develop a system for learning assembly programs based on segmenting observations of the demonstrator's hand into action primitives [102, 101]. The task execution is planned using instantiations of these primitives as logical building blocks with pre- and post-conditions.

Voyles *et al.* use primitives to interpret demonstration by segmenting observed actions into recognizable parts. By using an abstract representation of sensorimotor behavior, they contend that they are reproducing the intentions of the demonstrator, which they define as "the underlying skills that produced the motions of the demonstrator as opposed to the observable motions themselves" [156]. Indeed, this work recognizes the value of separating the underlying intention from the observable motor artifacts it creates.

There has been research on temporal segmentation of manipulation tasks into subtasks that can be characterized according to the type of grasp being used [95, 96]. Friedrich *et al.* describe a system where the demonstration is segmented into a sequence of primitives using a time-delay neural net [57, 56]. The universe of primitives was defined *a priori* for the domain of object manipulation. The inferred sequence of primitives is pruned by querying the user to establish their intent, and then parsed into a higher level planning representation. The planning system forms the basis for execution of new task instances.

Many approaches have taken a probabilistic approach to task representation, using motor primitive activations as a discrete event sample space. A discrete probabilistic representation such as Hidden Markov models (HMMs) can be used to represent the task using these events [129, 158]. This approach has been used to model an "egg-frying" task by creating motor primitives based on the tension signal from finger tendons [129], position and velocity trajectory following tasks [158], peg in hole insertions [81], and a block sorting task with a mobile robot [138], among others. Primitives can also serve as the basis for motion classification using a Bayesian inferencing framework [47, 46]. A related approach is to represent the task model as a latent random variable that must be inferred from observation [126].

Miyata *et al.* describe a system that optimizes motion captured trajectories for use in the robot using the null space projection of subordinate controllers to maximize secondary objectives [115]. Although the procedural abstraction is limited, this approach begins to decouple the explicit training signal of the demonstrator from the declarative aspect of the task; in this case, lifting an object onto a shelf. By abstracting the declarative task space, the robot can choose to optimize various secondary objectives while still accomplishing the task goal.

A more abstract declarative approach can use the natural hierarchical composition of tasks to structure the inference process [126, 43]. Some researchers have proposed a graphical approach: the task is represented as a (hierarchical) collection of distinctive subgoals with precedence relationships. Nicolescu and Matarić represent tasks for a mobile robot as a graph where the nodes represent behavior with pre- and post-conditions [124]. Pardowitz *et al.* represent tasks in the form of precedence graphs, where subgoals are achieved by action primitives and the precedence structure can be learned incrementally [125].

Dillmann describes a programming by demonstration system where knowledge about the task is represented using a grammar of symbols that are the elementary actions available to the robot [43]. These elementary actions are either hand-coded or assembled from basic sensor-based control laws. For generalization, they represent higher level subtasks in terms of "macro-operators" that are parameterizable action sequences for accomplishing certain task goals.

An implicit assumption of all these approaches is that the trajectory of the demonstrator is a key feature of the demonstration that is important for completing the task. I contend that this approach to imitative behavior will ultimately prove unsuccessful at recreating complex, reusable behavior. Aside from a narrow range of tasks that require specific trajectories, such as dance moves, pole balancing, or ball-in-cup tasks, the demonstrated trajectory is a single example of a deeper intentional principle underlying behavior. Demonstrations are merely samples of motion trajectories drawn from a distribution of possible solutions to a class of tasks.

As task representation becomes more abstract, the intentions are highlighted as opposed to motion artifacts. By decoupling the method for achieving a goal from the goal itself, the robot can separate perhaps inconsequential procedural details of the demonstration from the intentions that underlie demonstrations. Thus, observations of the activity of others can be used to infer generative models of the learner's own behavior. This allows the robot to achieve that goal using any means at its disposal.

2.3 The Teleological Approach

In the preceding discussion, I described various approaches for implementing the procedural and declarative aspects of a programming by demonstration system. Imitative

learning requires that not only must the observer find correspondence between the morphologies of the demonstrator and the observer, if they exist, but also understand the intention of the actions observed. The observer must infer the distinguishing cues in the environment and how they map to observed actions and the corresponding control decisions.

The teleological stance is proposed by Gergely *et al.* to explain the inferential abilities of infants [67, 37, 63]. It forms a teleological (as opposed to causal) relation among goals, means, and constraints of actions, linked by a principle of rational action. In order to be a valid teleological explanation, the observed action must be a rational means of achieving a goal given the physical constraints of the environment [35, 36, 37, 63, 65, 66, 67]. That is, a teleological explanation of observed behavior is valid if the action is the most efficient and rational way to achieve the specified goal given the constraints in the run time context.

Gergely *et al.* [64] provide an example of this perspective in which they reproduce an experiment originally performed by Meltzoff [112]. In the original study, 14-month old infants watched a demonstrator seated at a table switch a lamp on and off by leaning forward and using their head to press a large switch. When the children were seated alone in front of the lamp they reproduced the novel head action. Gergely *et al.* reproduced this experiment, except the demonstrator was visibly unable to use their hands to perform the head action, as they were wrapped in a blanket [64]. Gergely *et al.* found that these children, when reintroduced to the lamp, used their hands—not the novel head action—to actuate the switch. According to the teleological hypothesis, the children were able to extract the goal of the action—actuating the switch—and inferred that the head action was due directly to observable constraints on the use of hands and was not a salient aspect of the demonstrated task. Since

the children did not have the same constraints as the demonstrator, they chose the most efficient action to achieve the goal available to them: namely, using their hands to switch the lamp. Furthermore, according to Gergely's hypothesis, the children in Meltzoff's study reproduce the novel head action because they infer from the demonstration that absent a clear constraint on the use of hands, the head action is the preferred way to interact with the lamp.

Gergely goes on to propose that an agent builds teleological representations by observing interactions with the world. These representations consist of three components: goals, means, and constraints. In my work, the observations of the world are described by three analogous components: end states, behavior, and physical context. These observed attributes can be converted into an internal, teleological representation if they adhere to the principle of rationality. This principle is represented as a function that maps observed attributes into valid teleological representations.

2.3.1 Goals

In the teleological representation, goals are the abstraction of physical end states in the world. The end state is a relationship between the robot and some feature in the environment brought about by action. The trajectory is an artifact of the demonstrator attempting to communicate end state goals to the pupil. These goals are the result of intentional action of the demonstrator. While the student and teacher may use vastly different models of the world to achieve their intentions, knowledge can be shared if it is grounded in the common language of observable goals. No knowledge of the other's models is required. Furthermore, by extracting the goals

of the demonstration and not merely the trajectories, the student can apply its own specialized knowledge to achieve the demonstrated goals in many different contexts.

Parsing a demonstration by examining goals partially eliminates the morphological correspondence problem, as the trajectories used by the demonstrator might have no correspondence to the student. But if the student can determine the goal of the demonstration, it can reproduce this in a way unique to its morphology and capabilities. In many cases the task being demonstrated can be successfully completed using many different trajectories and even many different types of effectors. It is the end state of the environment after the task has completed—the goal—that defines success.

Consider a bimanual robot such as Dexter being shown how to perform a pick and place task. The objective is to move an object to a given location. The robot may use either arm to perform the task, and in fact this will result in very different trajectories, but the task will be successful as long as the object is moved to the target location. In this thesis, I take a teleological approach to imitation: the important features to be reproduced are not the trajectories used by the demonstrator, but the intended goal that produced those samples of possible trajectories. Furthermore, the features that are associated with the convergence of controllers are associated with goals.

A simple instantiation of a pick and place task can be represented by a sequence of reaching and grasp-related controllers: reach, grasp, reach, release. The convergence of a reach controller signals the robot must begin a tactile action (grasping or releasing an object), and it is the sensory references at this instant that the robot can associate with a goal state.

Gergely *et al.* do not address uncertainty in the observations that a student makes of a demonstration; they assume the agent can accurately parse the observed actions into meaningful segments. Roboticists are not so fortunate, and must cope with the large amount of uncertainty that arises when dealing with the physical world. In this dissertation, I have a much more focused domain for the robot, and assume that a visual processing system is capable of detecting and segmenting visual features that are relevant to the observed tasks.

2.3.2 Means

In the teleological stance, the means represent ways in which end states can be achieved. For example, in the lamp-switching experiment, hand and head movements are both plausible means of actuating the lamp switch. Others, feet, for instance, may also be possible in certain contexts. In the robotics domain, the means refer to the sensorimotor resources that are available and appropriate for successfully executing a behavior. In the behavioral framework described in this thesis, resource allocation is determined based on the procedural requirements of the behavior being executed. For example, in a pick and place task, Dexter must specify which arm to use for the pick and place subtasks. Note that these could differ if the object is transferred between hands.

2.3.3 Constraints

Closely aligned with means are constraints. In the teleological context, constraints are used to inform the selection of the appropriate means for a given action. In Gergely's version of the lamp-switching experiment, the blanket wrapped around the hands of

the demonstrator implied that they too constitute a viable means for actuating the switch when constraints on their use are not present.

Inferring constraints has to do both with interpreting the salient aspects of a demonstrated task and with the prerogative of the robot for selecting other solutions that represent its preferences at run time. The constraints placed upon the robot are the aspects of the environment that the robot must sense in order to successfully complete the task. The environmental context at run time forms the constraints that act on the robot in action selection. For Dexter, this includes detecting the relevant state of the environment, such as the location of graspable objects, as well as obstacle and collision avoidance.

2.3.4 Action Selection

The principle of rationality is the concept used by Gergely to describe the necessary conditions for observed attributes of actions to become a valid teleological description [67, 65, 37]. This principle states that the agent performs the most efficient action to achieve the desired end state given the physical context in which it is acting.

In this thesis, I assume that the robot is not starting from a *tabula rasa*, but instead has been endowed with a set of prior, parameterizable behaviors that it can use as a basis for interpreting observations. These behaviors are the sensorimotor framework used to form teleological representations. These consist of well-formed sequences of actions that are parameterized by the features of objects in the environment. The closed-loop controller is used to discover end states that are relevant to the robot, and it uses the constraints present in the environment to affect its allocation of effector resources.

Action selection using maximum likelihood or maximum *a posteriori* methods is analogous to the principle of rationality, assuming that the model was built on data collected from demonstration (which is assumed to be rational). Furthermore, the actions available to the agent are well-formed sequences of controllers that result in meaningful action. Therefore if the robot able to correctly parse its observations, it will by default have formed rational theories of action.

2.4 Biological Motivation

A human's remarkable ability to learn by imitation provides a source of inspiration for much research in humanoid robots [140], and provides motivation behind the approach taken in this thesis. There is an ongoing debate within the behavioral science literature about whether any species beyond humans and the great apes are capable of learning by imitation [24, 155]. In this section, I discuss some of the areas of overlap between the neurologic and cognitive processes hypothesized to be behind imitation, and how they relate to the framework presented in this thesis.

Mirror Neurons

Much is still unknown about the neurologic basis of imitation learning in humans and non-human primates, but *mirror neurons* are hypothesized to play a role in the process. Mirror neurons are a type of neuron in both humans and non-human primates that have an intriguing property: they fire both when an action is being executed and when that action is observed [135, 21, 100, 22]. Originally discovered using direct neuron recordings in macaque monkeys, functional magnetic resonance imaging (fMRI) studies have produced evidence of their existence in humans [84, 83, 55].

Furthermore, this empirical data suggests that mirror neurons are used not only for action recognition and execution, but also for understanding intention and goal-directed behavior [135, 54, 83, 55]. Mirror neurons provide clues as to how imitation learning, intent recognition, and other skills related to social behavior may work at the level of neuroanatomy.

In one such fMRI study, Iacoboni *et al.* found that a subset of "logically related" mirror neurons in the inferior frontal cortex fire in response to motor acts that are likely to *follow* an observed action [83]. Different sets of neurons were activated depending on the context of the observed actions. This suggests that mirror neurons play an important part in the process of recognizing intention. Additionally, the mirror neuron system represents an efficient and efficacious reuse of neural pathways that could provide the neurological substrate necessary for teleological reasoning.

This suggests that a way to implement a computational teleological framework is to use generative models that allow for both recognition and execution. To this end, I use closed-loop controllers as the basis for a mirror neuron-like behavioral substrate for the robot. The dual use capability of controllers for recognition and execution parallels the biological function of mirror neurons in humans. Furthermore, the selectively of the neurons' activity to the context of the action speaks to their involvement in the recognition and execution of functional goals with objects.

Affordances

Affordances are functional relationships between agents and the environment that are characterized by perceptual features [69]. Recent neuroimaging studies have shown that merely presenting tools to humans stimulates brain regions associated with ac-

tions [34]. This suggests that action-related information about objects is represented subconsciously when the tool is observed, giving support to the affordance-based view of action observation.

Affordances provide a natural categorization of objects based on function rather than appearance, which may vary drastically among similar objects. In the context of learning pick and place from demonstrations, consider grasp affordances: the ways to grasp an object in order to achieve a particular function [51, 5]. Some researchers hypothesize the existence of micro-affordances that correspond to specific types of grasps, such as power or precision grasps [49]. For example, a coffee mug has at least two distinct grasp affordances: one for drinking (typically by using the handle), and another for transporting. Affordances are behavioral properties that have associations with auxiliary characteristics of objects, e.g., visual appearance.

Previous work has examined robot tasks that can be described in terms of affordances [104], and in particular, when tasks can exploit tools [145]. De Granville *et al.* developed a representation for grasp affordances using parametric distributions over hand orientation [39]. In Chapter 6, I discuss a computational approach to the related problem of grasp prototype representation using probabilistic models that allows multiple objects to share a set of grasp prototypes.

CHAPTER 3

THE CONTROL BASIS AND SENSORIMOTOR SCHEMAS

One of the main contentions of this thesis is that a robot with a comprehensive set of built-in behavioral abilities is better able to learn from demonstration. The PbD framework described in this thesis requires that the robot be able to interpret its sensors in order to recognize end states, actions, and physical contexts. The foundation of this inferential process is built from closed-loop controllers: structures that output a control signal to a plant based on a reference and feedback from the plant's output, as shown in Figure 3.1. A closed-loop controller generates actions that achieve the goal state by suppressing errors and perturbations in a consistent way. In the teleological framework, end states correspond to conditions where a controller is converged. By activating a controller to execute motions, the robot behaves teleologically by seeking explicit goal states, under the interaction dynamics of the controller and the environment. A controller expresses an on-going relationship with the world: convergence indicates that the plant has reached a low error state with respect to the possibly time-varying reference. In this chapter, I describe the *control basis* approach [31] for organizing a collection of closed-loop controllers into a behavioral framework.

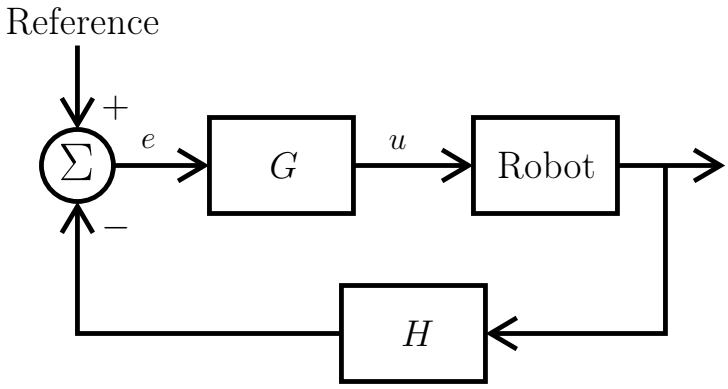

Figure 3.1. A closed-loop controller. The function G transforms the error signal e into a control signal u for the robot. The function H measures the output of the system and produces a sensor signal used to compute the error signal.

3.1 The Control Basis

The control basis is a combinatoric framework for constructing closed-loop controllers for the robot's lowest level motor units: second order controllers for position and force [30, 28]. The elements of the control basis consists of four finite sets:

- Σ is a set of physical or abstract sensors that generate signals (e.g. encoder, load cell, camera, etc.),
- Ω is a set of convolution operators that act on signals,
- Φ is a set of artificial potential functions, and
- T is a set of motor units that actuate degrees of freedom of the robot.

A closed-loop controller is configured by selecting a single element from each set.

Definition 3.1 (Controller)**.** *Let* $\mathsf{C} = (\sigma, \omega, \phi, \tau)$*, where* $\sigma \in \Sigma$*,* $\omega \in \Omega$*,* $\phi \in \Phi$*, and* $\tau \in T$*, define a controller in the control basis.*

In this work, I denote controllers with capitalized variables written in a sans-serif font ($\mathsf{A}, \mathsf{B}, \mathsf{C}, \ldots$).

Let $\mathsf{CB} = \{\Sigma \times \Omega \times \Phi \times T\}$ be the set of all controllers in the control basis.

Feedback elements are defined by the Cartesian product of sensors and convolution operators $f : \Sigma \times \Omega \rightarrow \mathcal{F}$.

Sensors $\sigma \in \Sigma$ generate signals in a specific coordinate frame $\mathcal{F}_\sigma$. For example, the position encoders of a robot publish the configuration $\boldsymbol{q} \in \mathcal{C}$ of the robot in configuration space: this is represented by the element $\sigma_{\boldsymbol{q}} \in \Sigma$. An abstract Cartesian position sensor $\sigma_{\boldsymbol{x}}$ publishes the workspace location $\boldsymbol{x} \in SE(3)$ of a given point on the robot using forward kinematics. The forward kinematics $\boldsymbol{x} = FK(\boldsymbol{q})$ can be defined using the Denavit-Hartenberg parameters of the robot [33]. A contact load cell on the fingertip of a robotic hand $\sigma_{\boldsymbol{f}} \in SE(3)$ may publish six axes of force and moment information of loads applied to the sensor.

Operators $\omega \in \Omega$ are convolution operators that act on signals published by sensors. Feedback is supplied by computing features $f : (\Sigma \times \Omega) \rightarrow \mathcal{F}_f$ from a signal using a convolution operator, specified in the coordinate frame $\mathcal{F}_f$. Feedback suitable for a position tracker can be obtained by applying a low-pass filter ω_{lp} to the configuration signal $\sigma_{\boldsymbol{x}}$, $f_{\boldsymbol{x}} = \sigma_{\boldsymbol{x}} \star \omega_{lp}$. The feedback derived from features drives the error suppressing behavior of closed-loop controllers. Note that in the following discussion, the (σ, ω) may be dropped from a feedback stream f if the sensor and operator pair is clear from the context.

Motor units $\tau \in T$ are embedded controllers for individual actuatable degrees of freedom of the robot. A motor unit represents the lowest level in the control basis and consists of an equilibrium setpoint controller on a single degree of freedom that accepts a reference value $\boldsymbol{u}_\tau$. Synergies of motor units, defined by a subset $\tau \subseteq T$ can also be controlled by higher level control basis controllers.

Artificial Potential functions $\phi \in \Phi$ are scalar mappings on the domain of the feedback signal $f(\sigma, \omega)$ that take an additional parameter specifying the goal reference of the potential, $\phi(f, \hat{f}) : \mathcal{F}_f \times \mathcal{F}_f \rightarrow \gamma$. The domain $\mathcal{F}_f$ is given by the coordinate system of the feedback signal $f(\sigma, \omega)$ associated with the controller. The feature stream $\hat{f}(\sigma, \omega)$ defines the references used to define the topology of ϕ. For notational simplicity, the feedback and reference arguments of ϕ will be dropped when they are clear from the context.

The feedback element $f(\sigma, \omega)$ and motor unit τ are abstract representations of the robot's sensing and actuating abilities, respectively. The artificial potential function ϕ is the glue that binds sensory information to robot behavior. The negative gradient of the potential field, $-\nabla\phi$ provides a greedy path to the minimum of the function. Assuming the artificial potential satisfies certain conditions, this minimum coincides with the goal reference $\hat{f}(\sigma, \omega)$. A potential function in Φ must satisfy certain conditions in order to be useful for robot control. The control signal resulting from an artificial potential field should exhibit asymptotic stability in the sense of Lyapunov [103]. This means that the field is smooth, it has a unique minimum, and the gradient is negative definite.

The locally linear relationship between changes in the motor reference $\boldsymbol{u}_\tau$ to changes in the feedback stream is given by the motor unit Jacobian matrix:

$$J_\tau = \frac{\partial f(\sigma, \omega)}{\partial \boldsymbol{u}_\tau}. \tag{3.1}$$

The controller acts to achieve the feedback reference $\hat{f}(\sigma, \omega)$ by descending the potential field ϕ. The relationship between changes in the feedback signal to changes in the potential field are described using the potential Jacobian

$$J_\phi = \frac{\partial \phi(f(\sigma, \omega), \hat{f})}{\partial f(\sigma, \omega)}. \tag{3.2}$$

By combining the motor and potential Jacobians, changes in the motor reference can be mapped to changes in the potential field by the controller Jacobian matrix:

$$\begin{aligned} J &= J_\phi J_\tau \\ &= \frac{\partial \phi}{\partial f(\sigma, \omega)} \frac{\partial f(\sigma, \omega)}{\partial \boldsymbol{u}_\tau} \\ &= \frac{\partial \phi}{\partial \boldsymbol{u}_\tau}. \end{aligned} \tag{3.3}$$

The dimension of J is $m \times n$, where $m = |\mathcal{F}_f|$ is the dimension of the feedback signal, and $n = |\boldsymbol{u}_\tau|$ is the number of degrees of freedom being controlled.

The pseudoinverse of the Jacobian is used to compute reference inputs to the motor units of a controller $(\sigma, \omega, \phi, \tau)$:

$$\Delta \boldsymbol{u}_\tau = -\kappa J^\dagger \Delta \phi, \tag{3.4}$$

where $J^\dagger$ is the pseudoinverse of J, and $\kappa > 0$ is a gain variable.

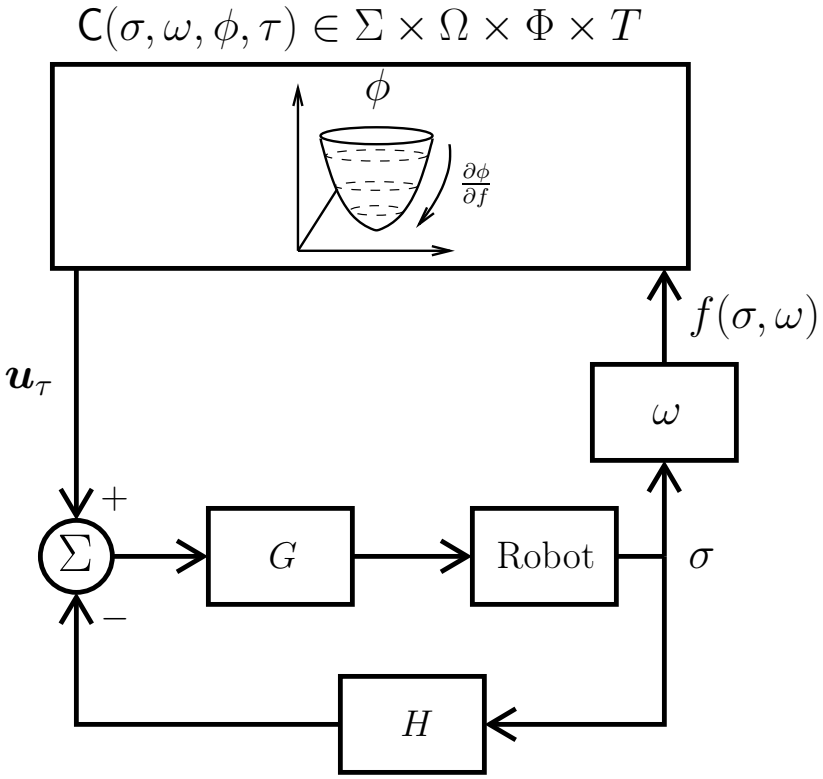

Figure 3.2. A schematic of the control basis. The controller C is composed from the basis sets of sensor, effector, and signal transforms. At the top level, the gradient of the artificial potential ϕ creates a reference signal $\boldsymbol{u}_\tau$ for effector resource τ. The underlying closed-loop controller actuates τ, which generates a sensor stream σ, that defines the current position in the potential field.

A diagram showing the relationship between the ϕ, σ, and τ and the resulting closed-loop controller is shown in Figure 3.2. The sets Σ and T are defined by the robot's physical sensors and effectors, respectively.

3.1.1 Artificial Potential Approaches in Robotics

Although artificial potential functions have been widely adopted for solving a number of robotics-related problems, they have been particularly successful when applied to the robot path planning domain. Path planning, that is, generating collision-free paths through configuration space, is a fundamental problem in robotics. The complexity of the problem grows exponentially with the number of DOF of the robot [25].

Khatib proposed using artificial potential fields for motion planning and joint limit avoidance in manipulator control [97, 98]. He defined a quadratic potential field in the robot's workspace that generated attractive forces towards the goal and repulsive forces away from obstacles and joint range limits. However, the artificial potential field was susceptible to local minima, which limited its effectiveness in complex environments.

One of the greatest difficulties in using potential fields for robot control is satisfying the condition of a unique minimum. Rimon and Koditschek defined a class of *navigation functions* that had many desirable properties for an artificial potential used for robot control [133, 99, 134]. In addition to having a unique minimum at the goal configuration, these functions must be continuously differentiable, have a finite gradient at all points, and have a non-singular Hessian at all critical points (a *Morse* function).

Another approach to constructing functional artificial potentials is to use harmonic functions. These are solutions to Laplace's equation, $\nabla^2\phi = 0$ that admit a unique global minimum. Harmonic functions ϕ_H describe many natural phenomena such as heat transfer and electrical potential in resistive networks. They have been applied to robot path planning, where they have the desirable property of minimizing hitting probability with respect to obstacles in the domain [32].

Artificial potential fields have also been used for force domain control problems. For example, in the area of robot grasping, by defining fields that minimize residual force and moment contact errors, the robot can generate grasps that fulfill force closure objectives [28, 128].

3.1.2 Multi-Objective Control

For controllers where $n > m$, where there are more controllable degrees of freedom than dimensions in the sensory feature space, then the system is said to be *redundant*, and the inverse may admit an infinite number of solutions. The Moore-Penrose pseudoinverse can be used to obtain the least squares solution to (3.4) in the n-dimensional space of $\Delta\boldsymbol{u}_\tau$ [118].

For humanoid robots that have many degrees of freedom, redundancy is a desired property that allows the robot to address multiple control objectives simultaneously. In the control basis, multi-objective controllers are formed by co-articulating ranked primitive controllers. If $\mathsf{C}_1 = (\sigma_1, \omega_1, \phi_1, \tau)$ and $\mathsf{C}_2 = (\sigma_2, \omega_2, \phi_2, \tau)$ are two controllers, where C_1 is the superior controller, then

$$\Delta \boldsymbol{u}_\tau = \mathsf{C}_2 \triangleleft \mathsf{C}_1 \tag{3.5}$$

$$= -\kappa_1 J_1^\dagger \Delta\phi_1 - \mathcal{N}_1 \kappa_2 J_2^\dagger \Delta\phi_2 \tag{3.6}$$

$$= -\kappa_1 J_1^\dagger \Delta\phi_1 - \left[I - J_1^\dagger J_1 \right] \kappa_2 J_2^\dagger \Delta\phi_2. \tag{3.7}$$

The multi-objective input (3.5)—read "C_2 subject to C_1"—is a projection of the inferior control objective C_2 onto the null space of the superior controller C_1 [118]. The null space operator $\mathcal{N}_1$ represents the locally linear approximation of the null space of the superior control objective. By projecting onto this subspace, the inferior controller does not destructively interfere with the higher-priority controller. The projection operation in (3.6) can be generalized to an arbitrary number of primitive controllers with different motor unit sets τ [118, 127].

3.1.3 Primitive Controllers

In this section I describe controllers constructed from the control basis by combining elements of $(\Sigma, \Omega, \Phi, T)$. Hart provides an extensive description of other controllers developed in the control basis [76].

3.1.3.1 A Controller for Reaching

Using the control basis, one can construct a simple but useful controller for reaching to a target specified by a feedback stream. The controller is specified as a tuple, $\mathsf{R} = (\sigma_{\text{stereo}}, \omega_G, \phi_U, \tau)$.

The feedback stream for the reaching target, f_t, is constructed by convolving the stereo images published by the cameras, σ_{stereo}, with a set of Gaussian filters, ω_G, to detect salient regions in the image plane to produce localized features in the workspace:

$$f_t(\sigma_{\text{stereo}}, \omega_G) \in \mathbf{R}^3. \tag{3.8}$$

Another feedback stream is the Cartesian location of the end effector, convolved with a Dirac delta that acts as an identity function:

$$f_{\boldsymbol{x}}(\sigma_{\boldsymbol{x}}, \delta) \in \mathbf{R}^6, \tag{3.9}$$

where $\sigma_{\boldsymbol{x}}$ represents the Cartesian location of a set of degrees of freedom represented by $\tau \in T$.

A quadratic potential function is a simple way to construct a control law that minimizes the error between the controllable variable $\boldsymbol{y} \in \mathbf{R}^N$ and a reference $\boldsymbol{r} \in \mathbf{R}^N$:

$$\phi_U(\boldsymbol{y}, \boldsymbol{r}) = \frac{1}{2}(\boldsymbol{y} - \boldsymbol{r})^T(\boldsymbol{y} - \boldsymbol{r}). \tag{3.10}$$

Following the gradient of this potential produces a linear path in Cartesian space to the target, as shown by the Jacobian

$$\frac{\partial \phi_U}{\partial \boldsymbol{y}} = (\boldsymbol{y} - \boldsymbol{r})^T. \tag{3.11}$$

The pseudoinverse of the combined Jacobian that maps from joint space of the manipulator to the potential field can be used to construct the control law

$$\Delta \boldsymbol{u}_\tau = -\kappa J_U^\dagger \Delta \phi_U(f_{\boldsymbol{x}}, f_t), \tag{3.12}$$

where

$$J_U = \frac{\partial \phi_U}{\partial \boldsymbol{x}} \frac{\partial \boldsymbol{x}}{\partial \tau}. \tag{3.13}$$

3.1.3.2 A Controller for Manipulability

Artificial potential functions have been widely researched in the robotics literature for addressing a number of problems related to kinematic conditioning. Manipulability is a conditioning metric that maximizes the manipulator's ability to sense and exert forces by keeping the Jacobian away from singular configurations. By including isotropic conditioning objectives in multi-objective control schemes, the robot can better respond to unexpected changes in the environment [119, 72]. Yoshikawa defined the *measure of manipulability* (MoM) field as

$$\phi_M(\boldsymbol{q}) = -\sqrt{|JJ^T|}, \tag{3.14}$$

where $J = J(\boldsymbol{q})$ is the manipulator Jacobian [161]. Although this field admits local minima, in practice it provides a useful metric for manipulator control.

The MoM field has been extended to other systems that can be described using a Jacobian. In an oculomotor system where the Jacobian transforms changes on the visual plane into changes in Cartesian workspace, then (3.14) defines a *measure of localizability* that maximizes the ability to perform stereo triangulation [74, 152].

By constructing a controller using the manipulability potential ϕ_M defined in (3.14), the robot can attempt to maximize the isotropic conditioning of a set of degrees of freedom. In the control basis, this controller is written $\mathsf{M} = (\sigma_{\boldsymbol{q}}, \delta, \phi_M, \tau)$. and the control law is

$$\Delta \boldsymbol{u}_\tau = -\kappa J_M^\dagger \Delta \phi_M(f_{\boldsymbol{q}}), \tag{3.15}$$

where the stream $f_{\boldsymbol{q}}$ are the configuration space coordinates of τ, and

$$J_M = \frac{\partial \phi_M}{\partial \boldsymbol{q}}. \tag{3.16}$$

Using the multi-objective control framework discussed in Section 3.1.2, the reach and manipulability controllers can be combined to form a composite controller that attempts to reach to a target location while maximizing manipulability as a subordinate objective:

$$\mathsf{M} \triangleleft \mathsf{R}. \tag{3.17}$$

3.1.3.3 A Controller for Grasping

Closed-loop controllers have also been developed for performing force-closure grasps on objects [128]. A precondition for this controller is that the object must be in the reachable workspace of the hand's fingers. The controller is defined as $\mathsf{G} = (\sigma_{\mathrm{f/m}}, \omega_{\mathrm{resid}}, \phi_U, \tau_{\mathrm{hand}})$, where $\sigma_{\mathrm{f/m}}$ is the stream of force and moment information from fingertip load cells, ω_{resid} filters the force and moment residuals, and τ_{hand} are the degrees of freedom in the hand.

3.2 Turning Controllers into Behavior

In this section, I outline how primitive controllers assembled from the control basis can be combined into more complex sensorimotor behavior. The control basis provides a combinatoric foundation from which to create controllers that seek to achieve given references in their feedback streams. In the following sections I describe a state representation based on the run time dynamics of control basis controllers. Using this state description, more elaborate behavior can be formed by combining single or multiple objective controllers in sequence.

3.2.1 Discrete State Dynamics

A controller assembled under the control basis suppresses disturbances and acts teleologically to achieve the goal state encoded in the artificial potential field. The dynamic response of the controller provides a natural way to represent the state of the robot with respect to its desired objectives. By classifying the response of a controller into a set of discrete states, the continuous, dynamic behavior of the robot can be characterized by a discrete state space. This discrete-event dynamic system (DEDS) representation of the control basis was first proposed by Huber [82].

The dynamic response of a closed-loop controller with a defined feedback signal can be described in terms of the instantaneous state of the potential field $(\phi, \dot{\phi})$ at time t. For asymptotically stable controllers, ϕ is positive definite and $\dot{\phi}$ is negative definite, so that the dynamic state $(\phi, \dot{\phi})$ remains in the lower right quadrant. Figure 3.3 shows the dynamic response of a closed-loop controller reaching multiple attractor states. The controller is quiescent when $\dot{\phi} < \epsilon$. The absolute level of error in the controller when in a quiescent state is described by ϕ. The quantity $\dot{\phi}$ represents the observed, finite-difference time derivative of ϕ.

This description of the dynamics of controller state was used by Coelho to learn a model of robot activity based on a history of task examples [29]. He derived a discrete state representation by matching the run-time trajectory to a set of exemplar trajectories.

For a given controller C_i at time t, define the quadripartite *convergence predicate* $\mathrm{P}_t(\mathsf{C}_i)$ that asserts whether the state has been measured, the absence of the feedback signal, and the convergence status of the controller:

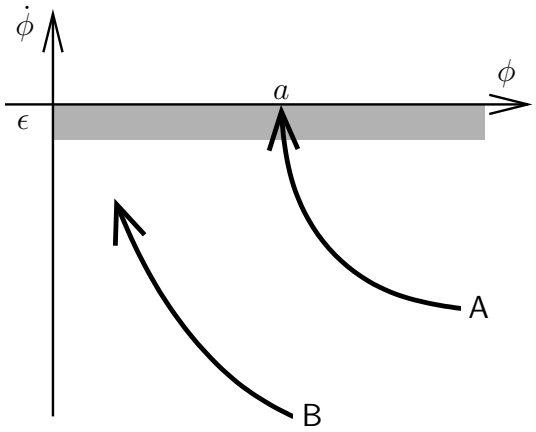

Figure 3.3. The dynamics of two controller activations. The grey area, $\dot{\phi} \leq \epsilon$, represents the quiescent state for the controller; when the dynamic state enters this region, the controller is considered to be converged. Controller A has reached a quiescent state where $\phi = a$. Controller B has not yet reached quiescence, but has already reached a lower absolute level of the artificial potential than A.

$$\mathrm{P}_t(\mathsf{C}_i) = \begin{cases} * & : & \phi_i \text{ state is unknown} \\ \varnothing & : & \phi_i \text{ has undefined reference} \\ 0 & : & |\dot{\phi}_i| > \epsilon \\ 1 & : & |\dot{\phi}_i| \leq \epsilon \end{cases}, \tag{3.18}$$

where ϕ_i denotes the value of the artificial potential at t, and ϵ is a positive constant. The state $*$ denotes that the status of the controller is unknown, and the state $\varnothing$ specifies that the reference $f_{\sigma,\omega}$ is not present in the feedback signal. Given the presence of the reference, then the states 0 and 1 denote whether the controller has reached quiescence. The transition dynamics between these four states are shown in Figure 3.4. The state $*$ represents an initial state: once a value other than $*$ has been evaluated, the predicate will only transition between the other three states.

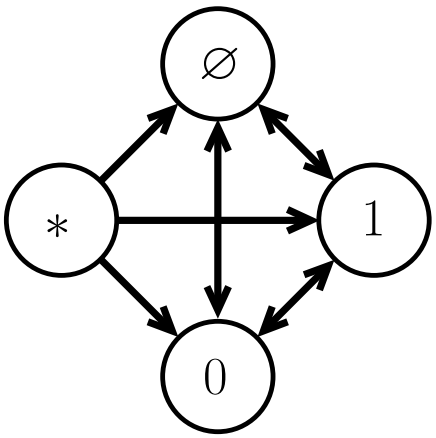

Figure 3.4. The quad state graph is a state machine describing the run-time dynamics of a control basis behavior. The starting state $*$ represents an undefined state in the underlying controller. The state $\varnothing$ denotes an undefined reference. The states 1 and 0 represent whether the controller has converged or not, respectively.

3.2.2 Sensorimotor Behaviors

Given a controller, the convergence predicate $\mathrm{P}(\cdot)$ provides a way to discretize its convergence status over time. In this section I describe how sensorimotor behaviors can be constructed by choosing a sequence of controllers to activate based on a state space constructed using the convergence predicate function. The behavior is a description of how a sequence of controllers should be executed in order to achieve convergence of a target controller.

Definition 3.2 (Sensorimotor Behavior)**.** *Define a sensorimotor behavior B as the tuple $(\mathcal{C}, \mathcal{Z}, \mathcal{G}, \mathcal{A}, \mathcal{T}, \pi)$ consisting of a set of controllers $\mathcal{C}$, a state space $\mathcal{Z}$, a set of goals states $\mathcal{G} \subset \mathcal{Z}$, an action space $\mathcal{A}$, a state transition model $\mathcal{T}$, and a policy π. Each controller $\mathsf{C} \in \mathcal{C}$ is a primitive controller. The action set consists of the controllers in $\mathcal{C}$ along with multi-objective combinations,*

$$\mathcal{A} = \{\mathsf{C}_i\} \bigcup \{\mathsf{C}_i \triangleleft \mathsf{C}_j \triangleleft \cdots \triangleleft \mathsf{C}_k\}. \tag{3.19}$$

Each state in $\boldsymbol{z} \in \mathcal{Z}$ is described by the vector $\boldsymbol{z} = [\mathrm{P}(\mathsf{C}_1) \ \cdots \ \mathrm{P}(\mathsf{C}_N)]$, where $N = |\mathcal{C}|$.

The policy is a mapping from state vectors to controllers, $\pi : \mathcal{Z} \rightarrow \mathcal{C}$, and determines the active controller for a given state. The set of goal states is strict subset of $\mathcal{Z}$, since a goal state corresponds to a quiescence condition for some controller in $\mathcal{C}$. Therefore, $\mathcal{G}$ excludes the state in which all of the controllers take the value $*$. The goal state represents the convergence of a specified set of controllers that represent the goal or *intention* of the behavior.

3.2.2.1 Search-Track

To make this discussion more concrete, consider some behaviors that have been developed for a humanoid robot. Hart *et al.* describe how these behaviors can be derived using a staged developmental process [77, 75, 76].

Consider a humanoid with an articulated vision system. The first behavior is known as "Search-Track," and consists of searching for a sensory stimulus and maintaining it in the center of the visual plane. This behavior uses two controllers:

Search $\mathsf{S} = (\phi_U, f_{\text{search}|\tau_v}, \tau_v)$. This controls the vision system's degrees of freedom τ_v using a quadratic potential to achieve references provided by $f_{\text{search}|\tau_v}$. This feedback stream samples "search" coordinates from a distribution $p(\mathcal{F}_{\tau_v})$ over the coordinate frame of τ_v that describes where interesting visual stimuli may be found.

Track $\mathsf{T} = (\phi_U, f_{\text{sat}}, \tau_v)$. The feedback stream $f_{\text{sat}} = (\sigma_v, \omega_{\text{sat}})$ filters out areas of highly saturated color from the raw visual stream σ_v. Controller T drives τ_v such that the visual stimulus is in the center of the visual plane.

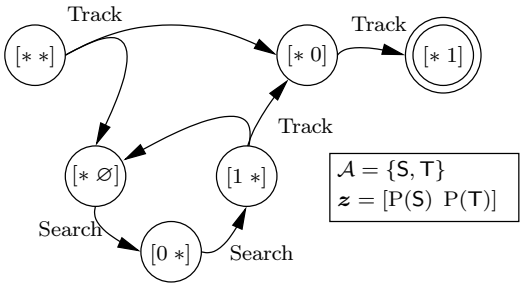

Figure 3.5. The Search-Track behavior. If a tracking reference is not found when the behavior begins, it activates the Search controller. This controller runs until it has located a tracking reference, at which point the Track controller is activated to foveate on the target.

Given this set of controllers, $\mathcal{C}_{\mathsf{ST}} = \{\mathsf{S}, \mathsf{T}\}$, the state space consists of all the possible values $[\mathrm{P}(\mathsf{S})\ \ \mathrm{P}(\mathsf{T})]$. The goal set consists of all the states in which the tracking controller is converged, $\mathcal{G} = \{[* \ 1]\}$. The policy for the "Search-Track" behavior B_{ST} is shown in Figure 3.5. From the initial state $[* \ *]$, the robot executes the track controller T. If no sensory stimulus is available to track, the behavior is in the $[* \ \varnothing]$ state, and the robot initiates the search controller S. Once the search controller has been activated, the robot servos the visual system to coordinates chosen according to the distribution $p(\mathcal{F}_{\tau_v})$. When S has converged on a search location, T is again activated. At some point when a visual stimulus is present, the behavior reaches the $[* \ 0]$ state signifying that T has a reference but has not yet converged. Once T has converged, the behavior reaches the goal state $[* \ 1]$.

If the multi-objective controller $\mathsf{S} \triangleleft \mathsf{T}$ is added to the action set, then an alternate form the behavior can be defined, as shown in Figure 3.6. In this formulation, the robot executes $\mathsf{S} \triangleleft \mathsf{T}$ until completion.

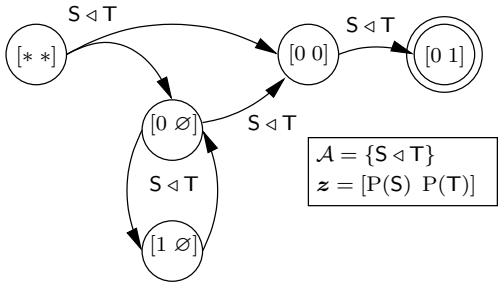

Figure 3.6. The Search-Track behavior using a multi-objective controller. In this instance, the Search and Track controllers are co-activated, and they are prevented from destructively interfering by using null space projection.

3.2.3 Hierarchical Structure

The concept of a sensorimotor behavior can be extended hierarchically by allowing behaviors to become sub-actions to other behaviors. Although each behavior is a temporally extended action, with its own state, it can be abstracted into a single action by extending the convergence predicate (3.18) for a behavior B as:

$$\mathrm{P}_t(B) = \begin{cases} * & : & \boldsymbol{z}_t = [* \ \cdots \ *] \\ \varnothing & : & \boldsymbol{z}_t = [\varnothing \ \cdots \ \varnothing] \\ 0 & : & \text{otherwise} \\ 1 & : & \boldsymbol{z}_t \in \mathcal{G} \end{cases} . \tag{3.20}$$

In the following section I describe a behavior that makes use of the Search-Track behavior as a sub-action.

3.2.3.1 Reach-Grasp

The ability to recognize objects of interest within the workspace is a prerequisite for manipulation tasks. Thus, the Search-Track behavior provides a useful starting point

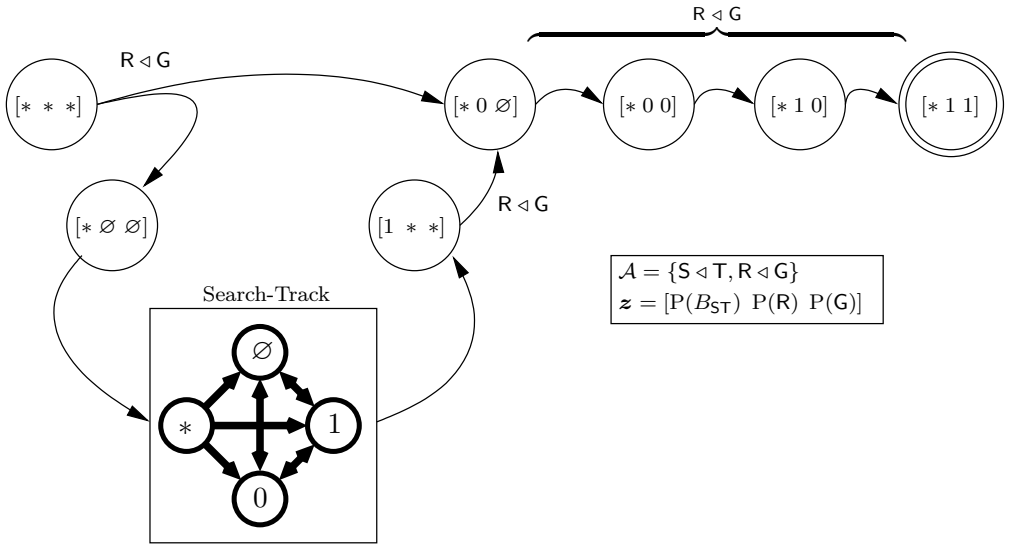

Figure 3.7. The Reach-Grasp behavior can initiate the Search-Track behavior if a reaching target is undefined. After Search-Track has found a target, the behavior executes the multi-objective Reach controller subject to a Grasp controller.

for developing hierarchical behaviors. By combining Search-Track with the Reach and Grasp controllers, one can construct a hierarchical behavior that identifies and attempts to grasp objects of interest within the workspace. This is the Reach-Grasp behavior B_{RG}.

In this behavior, the set of controllers is $\mathcal{C}_{\mathsf{RG}} = \{B_{\mathsf{ST}}, \mathsf{R}, \mathsf{G}\}$, and the state vector is $[\mathrm{P}(B_{\mathsf{ST}})\ \mathrm{P}(\mathsf{R})\ \mathrm{P}(\mathsf{G})]$. The goal state for this behavior has the grasp controller in the converged state. The Reach controller is configured to use the same feedback element as the Search-Track controller, for example, f_{sat}, will identify objects that have highly saturated colors.

Figure 3.7 shows the policy for the Reach-Grasp behavior, which was learned from a process of exploration and accommodation, detailed by Hart [76]. From the initial state, the robot assesses the convergence of the multi-objective controller $\mathsf{R} \triangleleft \mathsf{G}$. If the object stimulus is visible in the workspace, then the behavior transitions to the

state $[* \; 0 \; \varnothing]$. However, if no stimulus is immediately present, the robot transitions to state $[* \; \varnothing \; \varnothing]$ and activates the Search-Track behavior B_{ST}. Once an object stimulus has been found by Search-Track, the robot executes $\mathsf{R} \triangleleft \mathsf{G}$ to reach to and grasp the object.

3.3 Representing Complex Behavior with Schemas

In the preceding sections, I have outlined how closed-loop controllers can be composed in a programmatic way using the control basis. These controllers are individual teleologic units of action that achieve specified references, and they can be combined into more complex sensorimotor behaviors by introducing a state space induced by the convergence predicate. In this section I describe sensorimotor *schemas*: declarative abstractions of sensorimotor behaviors that admit multiple instantiations into executable behavior using procedural resources induced by run-time constraints. The schema is a generalization of the sensorimotor behavior described above, and requires decomposing controllers into declarative and procedural components.

3.3.1 Declarative and Procedural Decomposition of Controllers

A controller $\mathsf{C}_i = (f_i, \phi_i, \tau_i)$ can be decomposed into separate procedural and declarative specifications, $p(\mathsf{C}_i)$ and $d(\mathsf{C}_i)$, respectively. The procedural specification of the controller consist of the feedback and effector resources allocated to the controller:

$$p(\mathsf{C}_i) \equiv (f_i, \tau_i). \tag{3.21}$$

The declarative *type* specification for the controller describes the artificial potential function, and the coordinate frames of the feedback element and effector resource associated with the controller:

$$d(\mathsf{C}_i) \equiv (\phi_i, \mathbb{F}_{f_i}, \mathbb{F}_{\tau_i}). \tag{3.22}$$

The typing constraints of the declarative component define an equivalence class of controllers in the control basis:

$$\mathcal{D}_{\mathsf{A}} = \{\mathsf{C}_i \in \mathsf{CB} \mid d(\mathsf{C}_i) = d(\mathsf{A})\}. \tag{3.23}$$

The set of controllers $\mathcal{D}_{\mathsf{A}}$ represent an *abstract action* that maintain the same declarative component as controller A.

The procedural component, determined by run-time conditions, defines the topology of the artificial potential function and the area of the robot's workspace that may be affected by the controller. The declarative component describes the type of coordinate system over which the potential is defined, and the properties of the potential itself. The reach controller from Section 3.1.3.1, $\mathsf{R} = (f_{\boldsymbol{x}}, \phi_U, \tau)$, can be decomposed into procedural and declarative components as follows:

$$p(\mathsf{R}) = (f_{\boldsymbol{x}}, \tau) \tag{3.24}$$

$$d(\mathsf{R}) = (\phi_U, \mathcal{F}_{\boldsymbol{x}}, \mathcal{F}_{\tau}). \tag{3.25}$$

3.3.2 Sensorimotor Schemas

A sensorimotor *schema* is a hierarchical, declarative control structure that is invariant to run-time context. Each schema is a generalization of a behavior B, into declarative and procedural policies.

Definition 3.3 (Schema)**.** *A sensorimotor schema S is the tuple $(\mathcal{C}, \mathcal{Z}, \mathcal{G}, \mathcal{A}, \pi, \psi)$. The components of the schema are based on a behavior B. $\mathcal{C}$ is the set of controllers from B, $\mathcal{G} \subset \mathcal{Z}$ is the set of goal states, π is the* action *policy $\pi : \mathcal{S} \rightarrow \mathcal{A}$, and ψ is the* procedural *policy $\psi : \mathcal{A} \times E \rightarrow \Sigma \times \Omega \times T$, where E is the environmental context.*

The state set is composed of convergence predicate vectors $\boldsymbol{z} = [\mathrm{P}(d(\mathsf{C}_1)) \ldots \mathrm{P}(d(\mathsf{C}_N))]$, for $\mathsf{C}_i \in \mathcal{C}$.

The abstract action set $\mathcal{A}$ is composed of the declarative components of $\mathcal{C}$:

$$\mathcal{A} = \{d(\mathsf{C}_i) \ \cup \ d(B_j)\}. \tag{3.26}$$

Because a schema is a declarative abstraction, it provides a way of representing teleologic behavior across many different operating contexts and run-time conditions. The procedural components of the controllers within a schema are determined by the robot at run-time based on the environmental context E. In Chapter 5, I describe how probability distributions $p(\Sigma, \Omega \,|\, \mathcal{A}, E)$ can be used to represent this procedural policy.

Like the sensorimotor behaviors they generalize, schemas provide a hierarchical representation of declarative knowledge. By allowing the use of sensorimotor behaviors as actions, the schema can leverage and re-use previously acquired behaviors that

address specific sensorimotor goals. Furthermore, the hierarchical structure allows for a more simple, abstract representation at the highest level. Hart describes how schemas may be learned through a developmental, intrinsically motivated learning process [76].

By executing the actions according to the transition model of the schema, the robot will act to achieve the schema's goal condition. The transition model may support a number of different paths to a goal state. This allows the schema to represent different contingent behavior. The realized path through the state space will depend on the run-time conditions experienced by the robot. For example, the schema may have behavior for addressing various error states that may occur during execution.

3.3.3 Pick-And-Place

In this thesis, I have focused on robot manipulation tasks that fall under the general description of "pick and place" tasks [106, 89, 101]. These sorts of tasks require the robot to identify and manipulate objects within its workspace. While the specifics of the pick and place behavior depend on the run time environmental context, many common object manipulation tasks fall under this description, such as sorting and stacking tasks.

At its most basic, the Pick-And-Place schema describes the actions of acquiring an object in the workspace, transferring the object to a new location, and then releasing the object at that location. The first two steps can be achieved using the previously described Reach-Grasp behavior and Reach controller, respectively. The final "placing" component requires the introduction of a new controller to the robot's repertoire. The Place controller $\mathsf{P} = (f_{\text{net}}, \phi_U, \tau_{\text{arm}})$ is used to achieve a net force reference for a

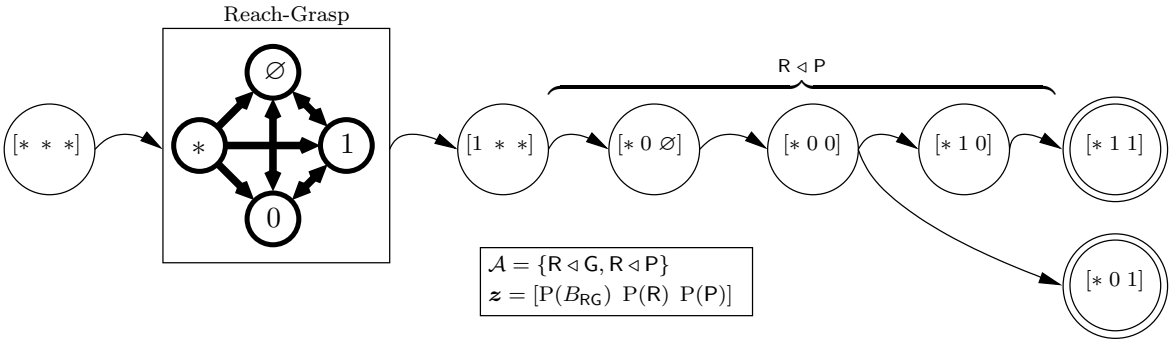

Figure 3.8. The Pick-And-Place Schema. The Reach-Grasp behavior is used to initially grasp the object, and then the multi-objective Reach subject to Place controller is executed.

grasped object by controlling the hand and arm resource τ_{arm}. It uses the quadratic potential, along with the net force stimulus $f_{\text{net}} = (\sigma_{\text{f/m}}, \omega_{\text{net}})$ that is derived from force/torque sensors in the hand.

Let S_{PP} be the pick and place schema. The set of behaviors in this schema is $\mathcal{C}_{\mathsf{PP}} = \{B_{\mathsf{RG}}, \mathsf{R}, \mathsf{P}\}$. The state vector $\boldsymbol{z} = [\mathrm{P}(B_{\mathsf{RG}})\ \mathrm{P}(\mathsf{R})\ \mathrm{P}(\mathsf{P})]$ is composed of convergence predicates for the Reach-Grasp, Reach, and Place controllers. Figure 3.8 shows the state transition graph for Pick-And-Place.

On initialization, the robot attempts to acquire the object through the use of the Reach-Grasp behavior. If it is successful, this puts the robot into the state $[1\ *\ *]$. From this state, the robot executes the multi-objective controller $\mathsf{R} \triangleleft \mathsf{P}$ in order to move the object to the correct goal location and controls the reaction force of the object while releasing it. The schema reaches a goal state when $\mathrm{P}(\mathsf{P}) = 1$, meaning that the place controller has converged.

The basic pick and place schema shown in Figure 3.8 can be augmented by multiple contingencies, for example, error recovery: if the object is inadvertently dropped by

the manipulator prior to reaching the desired location, the robot can regrasp the object and finish the original place phase. For bimanual humanoids, the schema can also contain contingencies for the case of pick and place locations which are reachable by different manipulators. In such a scenario, the robot must transfer the object between manipulators to successfully complete the behavior. These elaborations illustrate the type of basic sensorimotor competence that can be encoded in a schema; a more detailed description of this schema is given by Hart [76].

3.4 Discussion

The control basis architecture presented in this chapter provides the robot with a way of creating, representing, and reusing sensorimotor behaviors. The schema construct is a hierarchical way to represent teleological behavior that can be decomposed into a general, static declarative component, and the dynamic, procedural component that is applied at execution time using environmental features.

In the following chapters, I describe my approach to PbD based on the assumption that the robot has been endowed with a basic level of behavioral competence. Sensorimotor schema constructed using the control basis mechanisms provide a way to represent this competency. The question of the origin of these behaviors is left to the individual designer. Although they may be designed by hand, many of the behaviors described in this chapter have been learned via a developmental learning process that incorporates intrinsic reward on multi-modal sensor input [76].

CHAPTER 4

A DECLARATIVE REPRESENTATION FOR PROGRAMMING BY DEMONSTRATION

The declarative aspect of a demonstration describes the task that is being observed as an abstraction of states. As discussed in the Section 2.2.2, a common representational approach in the literature is to focus on the trajectory of the demonstrator. In this chapter, I argue that trajectories are an insufficient means for learning declarative knowledge from demonstration.

Trajectory-based approaches suffer from a lack of generalizable knowledge. A trajectory is valid only for a given environmental context. Similar trajectories may serve drastically different task objectives, and a large portion of the trajectory is typically not relevant to the task.

Instead, the robot should extract the *intention* of the demonstrator. Intention is a much more powerful property of the demonstration that allows the robot to abstract out a specific run-time context. The robot has a built in set of behaviors, outlined in Chapter 3, at its disposal. In this chapter I describe how the robot can use statistical inference along with its own set of behavioral schemas to interpret the information presented in a demonstration. The declarative information encapsulated in the schemas provide a framework for extracting intentions from the demonstration.

A demonstration session provides a continuous stream of high-bandwidth information to the observer. Limited computational resources means that the robot must pick and choose which parts of the demonstration merit full attention. While observing a teaching episode, the robot should extract the relevant teaching cues from the sensor stream in order to create a compact representation of the knowledge being expressed by the teacher. The teleological argument provides a motivation for deciding which portions of the teaching signal require additional scrutiny: information associated with end states should be inherently more interesting to the robot. The temporal structure of end states, signaled by the convergence of a controller, can "explain" the intent of a demonstration.

When an end state is observed, the robot records and associates the reference of the converged controller (and predicted schemas) with features that are spatially related to the reference. The robot can later create a segmentation of feature space based on the features associated with end states.

4.1 Declarative Representation of Schemas

The underlying process dynamics of a sensorimotor schema are described using a discrete set of states defined by the convergence status of the constituent closed-loop controllers. In this chapter, I describe how the robot can use this DEDS representation, along with a set of behavior schemas to interpret the actions of a demonstrator.

By matching the demonstration to one of its internal models of behavior, the robot explains the observed behavior using constructs it can use to recreate it. A further benefit of this approach is that by representing the demonstration as a sequence of schema executions, when the robot attempts to apply the knowledge gained from

demonstration, it can use all of the knowledge embedded within the schema—even if this was never observed in demonstration.

The schema matching process performs a type of behavioral quantization: the demonstrator's behavior is mapped onto prespecified declarative structures. In order to maximally capture the intention of the demonstrator, the basis set of declarative schemas must span the space of behavior that the robot is likely to observe.

Given a declarative specification, the robot uses the run time context to determine the appropriate procedural details required to execute the schema. These procedural details include proper resource allocation, and the identification of appropriate reference values for the underlying controllers. In Chapter 5, I describe how the robot can create probabilistic models of procedural context for a given schema conditioned on the run-time environment.

4.1.1 Hidden Markov Models

The Hidden Markov model (HMM) is an established method for representing a stochastic process that exhibits serial dependence between observations [131]. These models have been used in a wide variety of applications, including speech recognition [132], and have been used in PbD applications [158, 81, 2, 1]. In the following I will give a brief introduction to hidden Markov models.

Consider a sequential stochastic process observed over T time periods. Assume that at each time period, the system is in an unobserved (hidden) state s_t, with the full state sequence given by $S = (s_1, \ldots, s_T)$. Each time period also has an observation

x_t, such that the full set of observations is $X = (x_1, \ldots, x_T)$. An HMM describes a joint probability distribution $p(X, S)$ over the set of states and observations.

In the HMM, the sequence of states has the *Markov* property, meaning that the distribution of the current state given the previous state is conditionally independent of the rest of the state history:

$$p(s_t \,|\, s_1, \ldots, s_{t-1}) = p(s_t \,|\, s_{t-1}). \tag{4.1}$$

Furthermore, the current observation depends only on the current state:

$$p(x_t \,|\, s_1, \ldots, s_t) = p(x_t \,|\, s_t). \tag{4.2}$$

The sequential stochastic process given by the distributions (4.1) and (4.2) describe a type of dynamic Bayesian network [131, 40]. This model is shown in graphical form in Figure 4.1. A graphical model uses a graph to represent a conditional probability distribution. The nodes in the graph are random variables and edges represent conditional probability relationships between variables.

The HMM is a generative representation of the stochastic process. A *generative* model is one that represents a joint probability distribution such that one may sample the modeled parameter.

The distribution (4.1) describes the likelihood of the system moving from one state to another. For each step in the chain, the next state is given by a distribution that takes a parameter θ_{s_t} dependent on the state s_t:

$$s_{t+1} \,|\, s_t \;\; \sim \;\; p(s_{t+1} \,|\, \theta_{s_t}), \tag{4.3}$$

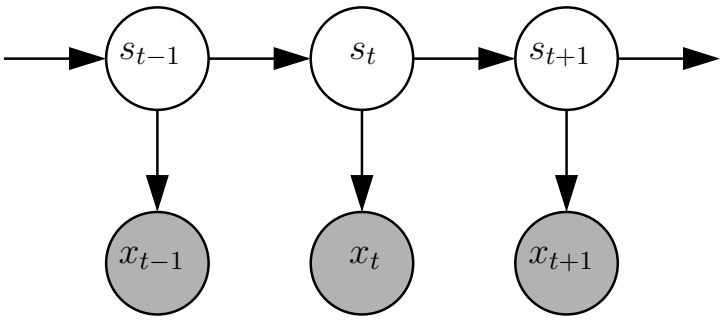

Figure 4.1. The hidden Markov model. The grey nodes are observed variables. At each time step, an observation is generated conditional on the latent variable. The next state variable is chosen from a distribution conditioned on the value of the current state.

and a distribution over the set of initial states:

$$s_1 \quad \sim \quad p(\zeta). \tag{4.4}$$

The set of parameters $\boldsymbol{\theta} = \{\theta_i\}$ and ζ describe the *transition* model for the process. If the full sequence of states is known, then its probability given a transition model is computed as

$$p(S \,|\, \zeta, \boldsymbol{\theta}) \quad = \quad p(s_1 \,|\, \zeta) \prod_{t=2}^{T} p(s_t \,|\, \theta_{s_{t-1}}). \tag{4.5}$$

At each time step, the system generates an observation according to the distribution (4.2):

$$x_t \,|\, s_t \quad \sim \quad p(x_t \,|\, \xi_{s_t}), \tag{4.6}$$

where $\boldsymbol{\xi} = \{\xi_i\}$ is the set of parameters that make up the *observation* model of the system. The probability of a sequence of observations generated from a known set of

states is given by

$$p(X \mid S, \boldsymbol{\xi}) \;=\; \prod_{t=1}^{T} p(x_t \mid \xi_{s_t}), \tag{4.7}$$

reflecting the conditional independence of observations on the state from which they were observed.

The joint distribution of the observations and states is described by

$$\begin{aligned} p(X, S \mid \boldsymbol{\theta}, \boldsymbol{\xi}, \zeta) &= p(X \mid S, \boldsymbol{\xi})\, p(S \mid \zeta, \boldsymbol{\theta}) && (4.8) \\ &= p(s_1 \mid \zeta)\, p(x_1 \mid \xi_{s_1}) \prod_{t=2}^{T} p(x_t \mid \xi_{s_t})\, p(s_t \mid \theta_{s_{t-1}}). && (4.9) \end{aligned}$$

The HMM model $\mathcal{H} = (\zeta, \boldsymbol{\theta}, \boldsymbol{\xi})$ is the tuple of parameters that make up the transition and observation models.

When trying to apply these models to real world processes, the goal is to identify the model $\mathcal{H}' \in \{\mathcal{H}_i\}$ that is most likely to have generated a sequence of observations X.

$$\mathcal{H}' \;=\; \underset{\mathcal{H}_i}{\operatorname{argmax}}\, p(\mathcal{H}_i \mid X). \tag{4.10}$$

The model $\mathcal{H}'$ is the *maximum a posteriori* (MAP) model, since (4.10) maximizes the posterior distribution $p(\mathcal{H}_i \mid X)$ given by Bayes' theorem,

$$p(\mathcal{H}_i \mid X) \;=\; \frac{p(X \mid \mathcal{H}_i) p(\mathcal{H}_i)}{p(X)}. \tag{4.11}$$

The likelihood term of (4.11) can be found by integrating over the entire state space,

$$p(X \mid \mathcal{H}_i) \;=\; \int_{\mathcal{S}} p(X \mid S, \mathcal{H}_i)\, p(S \mid \mathcal{H}_i) dS. \tag{4.12}$$

Naïvely computing (4.12) by iterating over every possible state sequence $S \in \mathcal{S}$ quickly becomes intractable as the number of observations grows. However, there exists a more efficient dynamic programming procedure known as the *forward-backward* method for computing this quantity in time $O(|\mathcal{S}|^2 T)$ [11].

The estimation problem for an HMM attempts to find the model parameters $\mathcal{H}_i$ with maximum likelihood for a given observation sequence,

$$\max_{\mathcal{H}_i} p(X \mid \mathcal{H}_i) \quad = \quad \max_{\mathcal{H}_i} \int_{\mathcal{S}} p(X, S \mid \mathcal{H}_i). \tag{4.13}$$

The Baum-Welch method is a well-known expectation-maximization algorithm for finding a local maximum of (4.13) [12, 42].

Example: HMMs for speech recognition

Hidden Markov models have been used with great success in the domain of speech recognition [131, 93]. In a typical application, the audio signal is broken into overlapping windows of 50 ms, spaced 30 ms apart [93]. Although the speech signal is nonstationary, the assumption is that at the short time scale of an individual window, the signal can be considered to be a sample from a stationary stochastic process. A feature vector is computed over each window in order to generate a more compact representation of the spectral properties of the window. Each feature vector is a single observation x_t, and the entire signal is represented as a sequence of observations $X = (x_1, \ldots, x_T)$.

The unobserved state s_t represents a stochastic process capable of generating short time scale spectral observations—each state is an individual speech phoneme. The

latent state sequence $S = (s_1, \ldots, s_t)$ represents a temporal relationship among the different phonemes. The HMM provides a way to model temporal relationships between these processes based on short term spectral observations.

In a system for single word recognition using HMMs, assume there is a vocabulary of W words to be recognized. Furthermore, assumed there are L labeled examples of each word, spoken by multiple speakers. The following steps are used to perform the recognition:

1. Using the Baum-Welch algorithm, build an HMM model $\mathcal{H}_w$ for each word in the vocabulary, $1 \leq w \leq W$, using the observations from the L examples of each word.

2. Given a set of observations X for an unknown word, compute the likelihood of each of the W models, $p(X \mid \mathcal{H}_w)$ using the forward-backward algorithm.

3. From the set of W likelihoods computed in the previous step, choose the word w^* whose model has maximum likelihood:

$$w^* = \operatorname*{argmax}_{1 \leq w \leq W} p(X \mid \mathcal{H}_w).$$

4.2 Interpreting Demonstration

In my approach to PbD, I assume that the observing robot has a set of sensorimotor schemas $\mathcal{S}$. The demonstration is viewed as a realization of a sequential stochastic process. I model the underlying process as an instantiation of a sensorimotor schema, and the robot must infer which schema $S \in \mathcal{S}$ is most likely to generate the observed actions. Taking a similar approach to speech recognition, I associate an HMM $\mathcal{H}_S$ with

each schema $S \in \mathcal{S}$. If HMM $\mathcal{H}_{S'}$ is the most likely model given the observations, then schema S' is considered to "explain" the demonstration. By abstracting the demonstration in terms of a schema, the robot has implicitly assigned a teleological goal to the demonstration that is addressed by the schema.

I will first present a simplified model of demonstration that assumes that each demonstration can be represented by a single schema instantiation. Later in this chapter I describe how this constraint may be relaxed.

Let schema $S \in \mathcal{S}$ consist of a set of states $\mathcal{Z}$ and actions $\mathcal{A}$. Each state $\boldsymbol{z} \in \mathcal{Z}$ is a vector of convergence predicates

$$\boldsymbol{z} = [\mathrm{P}(d(\mathsf{C}_1)) \ldots \mathrm{P}(d(\mathsf{C}_n))]\,. \tag{4.14}$$

A *monitor* is a special type of controller defined with no effector resource: $\underline{\mathsf{C}} = (\sigma, \omega, \phi, \varnothing)$. A monitor lacks the ability to effect the environment; instead, it updates its potential function based on the actions of other controllers.

For each abstract action $d(\mathsf{C}_i)$ in S, one can define a *monitor set*:

$$\underline{M}(\mathsf{C}_i) = \{\underline{\mathsf{C}_j} \,|\, d(\underline{\mathsf{C}_j}) = d(\mathsf{C}_i)\}. \tag{4.15}$$

Each element of $\underline{M}(\mathsf{C}_i)$ is a monitor with a different feedback reference. Consider a schema that contains the abstracted version of the search-track controller, $\mathsf{S} \triangleleft \mathsf{T}$. The monitor set $\underline{M}(\mathsf{S} \triangleleft \mathsf{T})$ contains multiple instantiations of search-track, each with a different stimulus reference. For example, a monitor set can be constructed using the saturated pixel blob feedback reference, f_{sat}, and the motion area reference, f_{motion}:

$$\underline{M}(\mathsf{S} \triangleleft \mathsf{T}) = \{\underline{\mathsf{S}(f_{\text{sat}}) \triangleleft \mathsf{T}(f_{\text{sat}})}, \underline{\mathsf{S}(f_{\text{motion}}) \triangleleft \mathsf{T}(f_{\text{motion}})}, \ldots\}. \tag{4.16}$$

For the purposes of observing a demonstration, the robot should have as many monitor instantiations per abstract action as possible, to span the space of possible stimuli that may be generated by the demonstrator. Like mirror neurons, those neurological structures that respond to both executing and observing certain types of actions, monitors allow behavior-producing schemas to be used to observe behavior. By associating an HMM with each schema, the robot can identify the schema most consistent with the demonstration.

Controller convergence can be evaluated in a monitor: convergence denotes whether the state of the robot is such that the controller would be converged *if* it were actuating the robot. The convergence status of a monitor set is an operator on the elements of the monitor set $\underline{M}(\mathsf{C}_i)$:

$$\mathrm{P}'(\underline{M}(\mathsf{C}_i)) = \begin{cases} 0 & : \quad \text{if all } \mathrm{P}(\underline{M}_k(\mathsf{C}_i)) = 0 \\ 1 & : \quad \text{if any } \mathrm{P}(\underline{M}_k(\mathsf{C}_i)) = 1 \end{cases}. \tag{4.17}$$

If the robot observes a goal condition for an instantiation of the abstract action, then a monitor in the set will be in a converged state.

Prior to beginning the demonstration, the robot creates monitor sets from the controllers in the union of the abstract actions represented in its universe of schemas $\mathcal{S}$. Let $\underline{\mathcal{M}}$ represent all the instantiated monitors:

$$\underline{\mathcal{M}} \;=\; \bigcup_{\mathsf{C}_i \in \mathcal{S}} \underline{M}(\mathsf{C}_i). \tag{4.18}$$

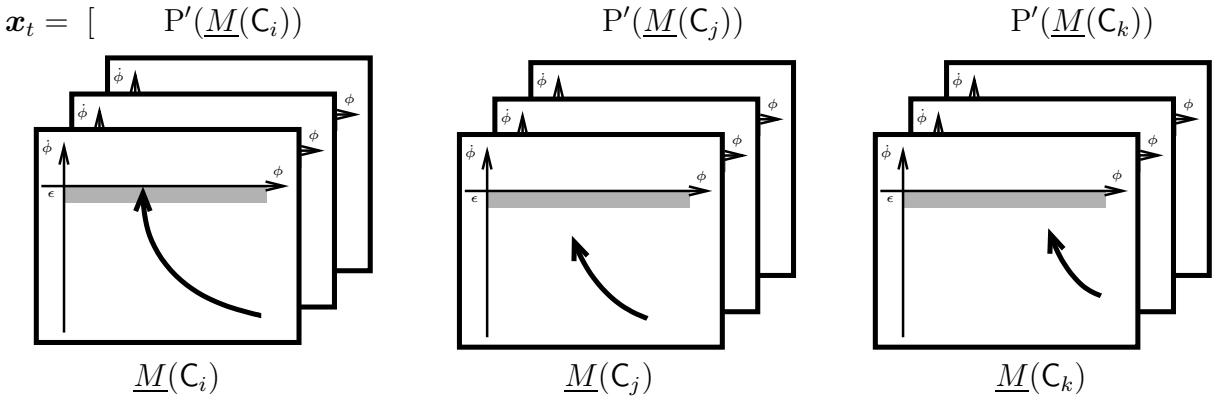

Figure 4.2. The organization of monitors into an observation vector. In this figure, there are three monitor sets for controllers i, j, and k. Each monitor set consists of a group of monitors with different controller references. The state variable $\boldsymbol{x}_t$ is composed of the concatenation of the convergence status of each monitor set.

The references for these monitors come from the set of available feedback elements $(\sigma, \omega) \in \Sigma \times \Omega$ in the control basis. These represent sources of feedback that encode stimuli of interest to the robot, based on its prior knowledge. Note that while the robot needs to instantiate enough monitors in order to span the space of expected observed behavior, it may also be susceptible to over-fitting if the monitors are too densely populated. Such a scenario may generate spurious monitor convergence, leading to a noisier signal.

At each time step in the demonstration, the robot constructs a single observation vector that consists of the convergence status of all monitor sets in $\underline{\mathcal{M}}$:

$$\boldsymbol{x}_t = [\mathrm{P}'(\underline{M}_1(\mathsf{C}_i)) \ldots \mathrm{P}'(\underline{M}_K(\mathsf{C}_i))],\ \underline{M}_j(\mathsf{C}_i) \in \underline{\mathcal{M}}. \tag{4.19}$$

The observation vector has size $K = |\underline{\mathcal{M}}|$, where K is the number of distinct abstract actions in the robot's universe of schemas. The organization of abstract actions into a single observation vector is shown in Figure 4.2.

In order to evaluate the likelihood of each $\mathcal{H}_S$, it is necessary to define each HMM's observation and transition models. For schema S, the transition parameters come from $\mathcal{T}$, the transition model of the schema, such that $\boldsymbol{\theta} = \mathcal{T}$. Since the schema's state space is discrete, a multinomial distribution can be used to model the transition probabilities between states:

$$s_{t+1} \,|\, s_t \;\sim\; \text{Multinomial}(s_{t+1} \,|\, \theta_{s_t}). \tag{4.20}$$

The observation model of $\mathcal{H}_S$ is a distribution over $\boldsymbol{x}_t$ given a latent state, $p(\boldsymbol{x}_t \,|\, \xi_{s_t})$. Because $\boldsymbol{x}_t$ consists of all the unique abstract actions in the robot's universe, each schema's state vector is composed of a subset of the abstract actions in $\boldsymbol{x}_t$. Therefore, it is possible to evaluate the likelihood of each schema's observation model given $\boldsymbol{x}_t$. Each state in a schema is defined by the convergence status of the underlying abstract actions in that schema. Therefore, state s_t can be considered a likely state for $\boldsymbol{x}_t$ if the abstract actions in s_t have the same convergence values in $\boldsymbol{x}_t$. This is represented in the following distribution:

$$p(\boldsymbol{x}_t \,|\, s_t) = \sum_{x \,|\, s_t = x} \delta_x, \tag{4.21}$$

where δ_x is a Dirac delta function at x. The atoms x are the realizations of the observation vector that have the same convergence status as the abstract actions in s_t.

After T time steps, the robot has a set of observations $\boldsymbol{x} = [\boldsymbol{x}_1 \dots \boldsymbol{x}_T]$, and can compute the most likely schema by maximizing the posterior distribution $p(\mathcal{H}_{S'} \,|\, \boldsymbol{x})$ using (4.21), (4.20), and the forward algorithm.

The robot also keeps a history of which monitors within each monitor set have converged, so that it can build a procedural model of the demonstration as well. This allows the robot to track which procedural details are being demonstrated by the teacher. The feedback elements of $\underline{M}(\mathsf{C}_i)$ that are converged provide information on the procedural elements of the demonstration. Knowing which feedback elements elicited a converged response during the demonstration allows the robot to build a procedural policy $\psi_{S'}$ for the inferred schema. The procedural policy determines the effector resources allocated to the schema based on the environment and preferences at run-time. A further benefit of this approach is that the robot can use the state transition model $\mathcal{T}$ of the inferred MAP schema S' to estimate which controllers may be activated next.

4.3 Applying Knowledge from Demonstration

After observing a demonstration, the robot infers the most likely schema that explains its observations. Given the declarative structure of this schema, the robot forms a procedural model for the schema based on the features exhibited by the demonstrator. Once the robot has acquired a declarative and procedural model, it can apply the knowledge learned from demonstration to new task instances. In this section I outline an experimental validation of the declarative inference process outlined above.

4.3.1 Experimental Apparatus

The experimental platform used in this dissertation is Dexter, the UMass bimanual humanoid, shown in Figure 4.3. Dexter has two manipulators—Barrett Technology's Whole-Arm Manipulator (WAM)—arranged in an anthropomorphic configuration.

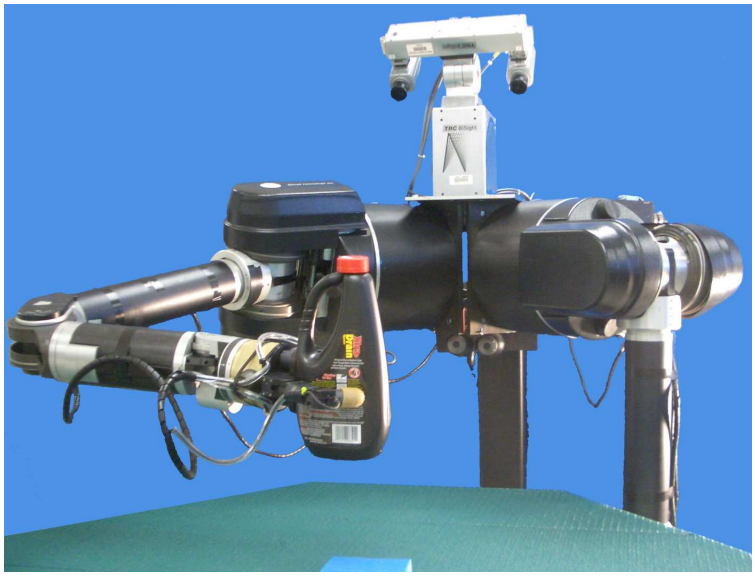

Figure 4.3. Each of Dexter's arms has 7 DOF and is equipped with a three-fingered hand with four total degrees of freedom. The stereo head has four degrees of freedom.

Each WAM has seven degrees of freedom with a cable transmission and direct drive motors. Additionally, each WAM is equipped with a three-fingered hand, also manufactured by Barrett, that features six-axis force/torque load sensors on the fingertips. Dexter has a stereoscopic vision system comprised of a shoulder-mounted pan tilt unit with four degrees of freedom (pan, tilt, and independent vergence for each camera).

The teleoperation system uses a glove-based controller worn by the operator. The glove is augmented with IR LEDs that are triangulated by a base unit. This allows the system to detect six degrees of motion of the teleoperator's hand. Further, the glove also measures finger flexion of all five fingers. Although Dexter has a stereo pair of cameras, the wide baseline between the cameras makes natural 3D video awkward for a teleoperator. Therefore, Dexter has a second stereo pair of video cameras with a narrower baseline mounted on Dexter's chassis under the shoulders. This allows the teleoperator to have a 3D view of Dexter's workspace.

Figure 4.4. The teleoperator's viewpoint. The icon in the upper right denotes whether the teleoperator is actively controlling the robot. The graph in the upper-left denotes the convergence status of the active schema. The red circle denotes the current state, while the dark red circles are previously visited states.

4.3.2 Experiment: A sorting task

For experimental validation of the declarative inference process outlined above, a pick and place task was demonstrated to Dexter. In this task, orange and blue objects were placed on the table in front of the robot. Two containers were placed behind the objects. The task was to place the orange objects into the right bucket, and the blue objects into the left bucket. In order to demonstrate the task, the teacher teleoperated Dexter's right arm, $\tau_{\mathbf{x,r}}$, and placed the orange object in the right bin, and the blue object in the left bin.

Figure 4.4 shows a screen grab of the video feed used by the demonstrator. The teleoperator has the option of viewing the scene in 3D using anaglyphic stereo. The

traffic light icon in the top right indicates when the teleoperator is actively controlling Dexter. The graph icon in the upper left of the frame is a representation of the robot's current inferential state. Each of the nodes represent a different state in the pick and place schema. Nodes colored gray are unvisited, dark red nodes are visited states, while the bright red node represents the current state.

After observing this demonstration, Dexter can infer the most likely declarative schema using (4.13). For this task, $\mathcal{H}' = S_{\mathsf{PP}}$, the pick and place schema. In Chapter 5 I describe how the robot infers a procedural model for the schema S_{PP} given the observed task. This model, M_{PP}, associates features of the objects in the task with the feedback elements of the controllers in S_{PP}.

Give the declarative model S_{PP} and the corresponding procedural model M_{PP}, the robot can apply the knowledge learned in demonstration to new task instances. After demonstration, Dexter is presented with objects and applies the knowledge from demonstration to place the objects into the appropriate buckets.

Once the pick-and-place schema has been identified as the "source" of the observations in the demonstration, the robot can leverage all the sensorimotor knowledge present in the schema. The schema encapsulates a policy for resource allocation based on environmental context. Figure 4.5 shows a sequence of video frames showing Dexter sort a blue object. Note that Dexter uses its left arm, $\tau_{\mathbf{x},\mathbf{l}}$ to perform this task, even though the teleoperator only performed the demonstration using the right arm. In this case, the pick-and-place procedural policy ψ_{PP} identified the left arm as necessary for executing the schema instance given the location of the blue ball.

Figure 4.6 shows another sequence of video frames where Dexter uses both arms to place the orange ball in the right bin. Dexter first grasps the ball using its left hand

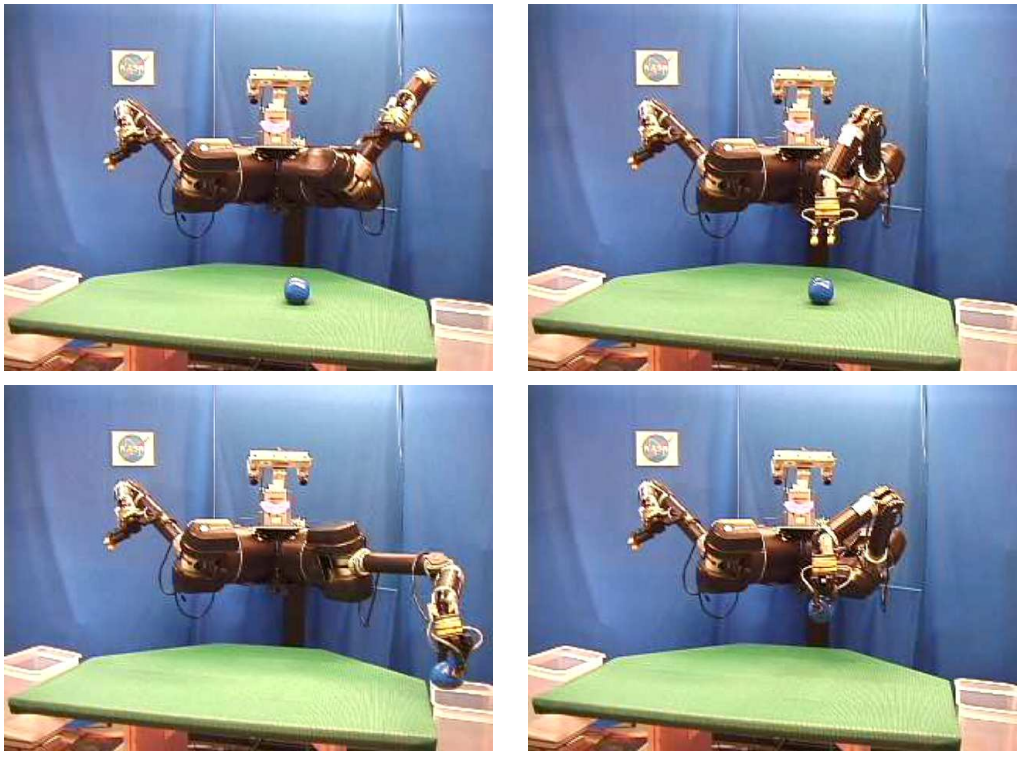

Figure 4.5. A sorting sequence, clockwise from top left. The sorting task is to place orange objects in the right bin, and blue objects in the other. The original demonstration used Dexter's right arm and bins in different locations. In this execution, Dexter autonomously uses its left arm to sort the ball into the left bin.

and places it on the table in the workspace of the right hand. After this, Dexter grasps the object with its right hand and places it in the bin. The pick-and-place schema S_{PP} has a contingency for performing this type of hand to hand transfer. Although this type of transfer was never performed by the demonstrator, since Dexter uses S_{PP} as a declarative representation of the demonstrated task, it can use any of the sensorimotor behavior contingencies available in the schema.

The declarative schema provides a teleological representation of the demonstration: the schema describes a state machine for achieving a behavioral goal. The schema is an abstract representation of the task, and the procedural policy allows Dexter to execute the schema in a variety of settings. Although the teacher may have provided a limited set of procedural examples, the procedural policy associated with the schema allows Dexter to generalize the demonstration to novel settings. The important aspects of the demonstration that must be extracted are the declarative goal—represented by the schema—and the necessary procedural details associated with closed-loop controller references used by the schema.

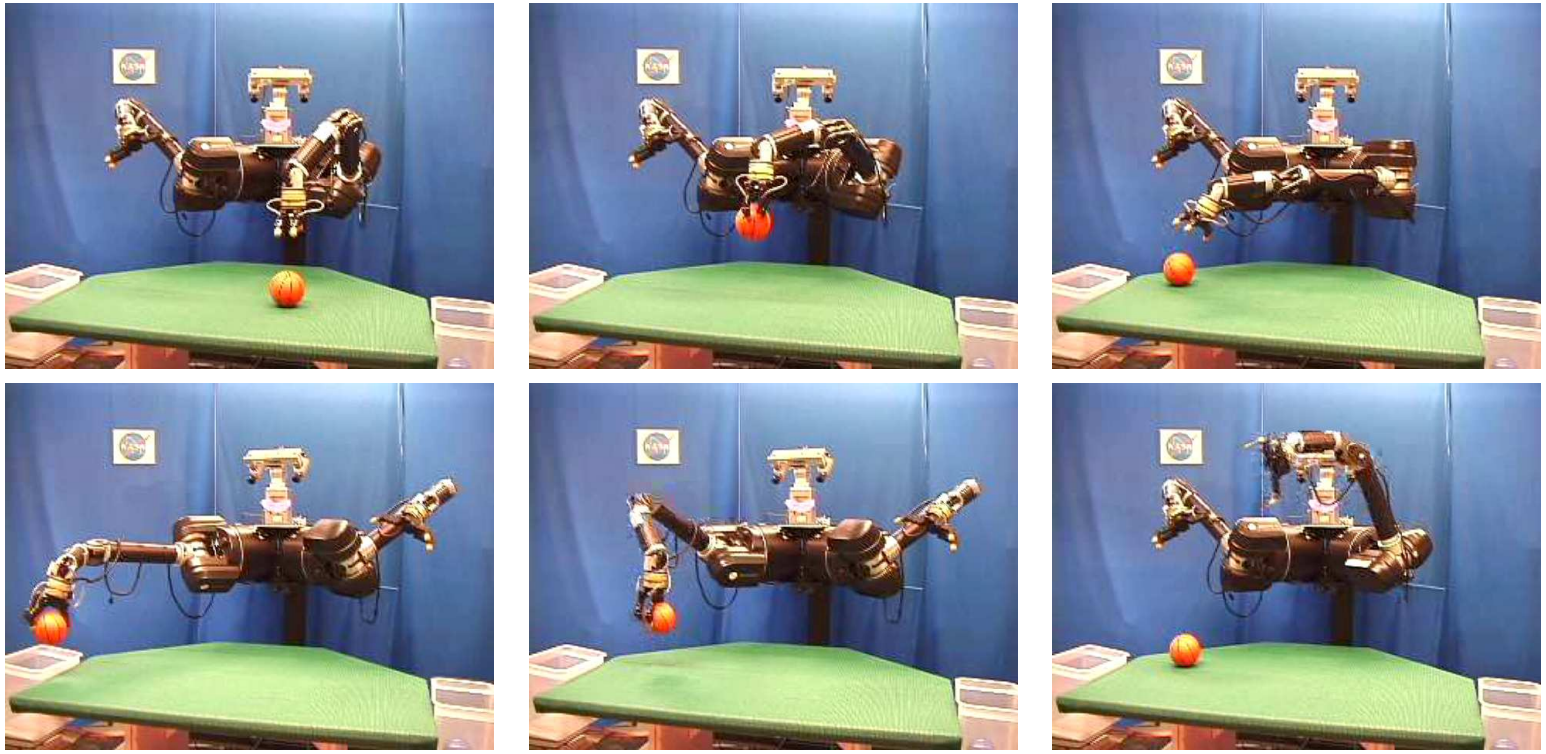

Figure 4.6. A sorting sequence, clockwise from top left. In this execution, the robot must sort the orange object into the right bin. This requires transferring the object between the workspaces of the two arms. This illustrates the contingency event of arm to arm transfer in the Pick-And-Place schema.

4.3.3 Discussion

The experiment discussed in this section provides a proof of concept of the declarative inference architecture presented in this chapter. Although the experiment is limited in scale, it shows the power of leveraging the schematic representation of demonstration. The robot is able to apply a range of knowledge that it has acquired to perform the task, even if that particular behavior was not demonstrated.

I described how the robot can assign a single schema as "responsible" for the observed demonstration. It is possible to extend this architecture hierarchically in order to describe tasks as sequences of schema executions. The hierarchical HMM is an abstraction of the HMM model that allows for multiple levels of latent variables [23]. Using hierarchical HMMs, one can assign separate schema invocations to different portions of a single demonstration.

CHAPTER 5

PROCEDURAL INFORMATION FROM DEMONSTRATION

In previous chapters, I have discussed a declarative representational framework that the robot uses to represent and infer sensorimotor behavior. The sensorimotor schema defined in Chapter 3 contains a *procedural policy*:

$$\psi : A \times E \rightarrow \Sigma \times \Omega \times T \tag{5.1}$$

that maps abstract actions and environmental conditions to the procedural assignment for a controller: the appropriate feedback element ($\Sigma \times \Omega$) and effector resource (T). The environmental context E consists of observed features and any task identifying information provided by the demonstrator.

In this chapter I introduce methods for inferring procedural policies from demonstration. I represent these policies using a generative, probabilistic model. I take a Bayesian approach, by building models that fully leverage the robot's prior knowledge. A benefit of the approach is that the robot can refine its model iteratively as new data becomes available. This is particularly useful in PbD tasks, as it allows the teacher to evaluate the robot's performance of the target task and reduce unnecessary training examples. Furthermore, prior information is useful in the inference process when observations are sparse.

Skill transfer is a term used to describe the ability of the robot to generalize concepts from one task instance to another. In this case, demonstration provides the information used to infer parameters for a probabilistic model. This model allows the robot to account for uncertainty in the training signal, and provides for generalization of the teaching patterns used by the demonstrator. The probability distribution represented by the model is used to evaluate the utility of actions under new operating contexts. The robot forms a policy using this measure of utility.

In the previous chapter, I described how the robot could infer a declarative "explanation" of the demonstration in the form of a sensorimotor schema. The schema provides a roadmap for identifying important features in the demonstration. These are features associated with the goal states of the controllers within the schema. The procedural policy is responsible for selecting the feedback elements and effector resources required to execute a schema, conditioned on the features in the environment.

The robot is awash in sensory feedback from the demonstration, and the schema provides a way to focus computational resources on the sensory information that provides the most information relevant to the task begin demonstrated. By using a schema mechanism to abstract the actions performed by a demonstrator, the robot is reducing the computational load during the observation phase, and disregarding information that is not relevant to successful completion of the task. Furthermore, the robot can use the knowledge it has been given up-front—sensorimotor schemas with generative Bayesian models of procedural context—to reduce the amount of instruction required to teach new task concepts.

In the remainder of this chapter, I describe how to represent the environment context used in the procedural policy, and how to extract it from the run-time sensor stream.

Then I discuss a probabilistic model that relates features in the environment to the procedural context required for schema execution. I examine in detail the example of the pick and place schema, and describe a feature model for that task. Then I discuss the results of experiments using that model for a sequence of pick and place tasks. I also discuss how the generative, Bayesian approach used by the model can improve demonstration performance and assist in incremental learning.

5.1 Generative Models for Procedural Policies

As described in Chapter 3, the procedural parameters of schema S are those that depend on the context in which the schema is executed. While the declarative structure of the schema determines the type and order of controllers that are executed, the procedural context specifies the run-time goals that are used to generate motions in each controller in the schema. These parameters are the feedback elements and resource allocations for the underlying closed-loop controllers in the schema.

A single schema may have multiple abstract actions, each requiring different feedback and effector resources. Let $A_S = \{a_1, \ldots, a_n\}$ be the abstract actions in schema S, $R_S = \{r_1, \ldots, r_n\}$ be control basis feedback elements $r_i = (\sigma_i, \omega_i)$, and $T_S = \{\tau_1, \ldots, \tau_n\}$ be the effector resources for S. In order to execute S, the schema must be proceduralized by selecting sets R_S and T_S to provide feedback and effector resources.

The types of feedback elements and effector resources that are used to proceduralize a schema depend on the abstract actions in the schema and the environmental context in which the schema is to be executed. Furthermore, there is a dependence between the abstract actions within the schema, and for most schemas, the robot can not proceduralize each abstract action independently. For example, in the pick

and place schema, the proceduralization of the abstract place action depends on the proceduralization of the preceding pick action.

Therefore, I represent the procedural policy as a joint distribution

$$\begin{aligned} \psi(S, E) &= p(a_1, \ldots, a_n, r_1, \ldots, r_n, \tau_1, \ldots, \tau_n, E) && (5.2) \\ &= p(A_S, R_S, T_S, E). && (5.3) \end{aligned}$$

Given the schema and environmental context, the policy can be written using Bayes' theorem as

$$p(R_S, T_S \,|\, A_S, E) = \frac{p(A_S, E \,|\, R_S, T_S)\, p(R_S, T_S)}{p(A_S, E)}. \quad (5.4)$$

The identity of the schema S is inferred using the declarative inference described in Chapter 4. The conditional probability structure encoded by the state and action space in the schema means that (5.2) is not a fully connected graph; instead, the exact relationship among the variables depends on the structure of the schema.

The declarative state transitions in the schema identified using the HMM model provide the robot with intermediate, teleological landmarks: end states of the constituent controllers in the schema. In this chapter, I describe a generative, Bayesian probability model the robot infers from demonstration to provide a procedural policy of the task.

I chose to use a generative, Bayesian feature model because it has properties that help reduce the amount of demonstration required for concept learning. Generative and discriminative approaches represent two different ways of modeling machine learning

tasks. It has been shown that asymptotically, as more data is acquired, a discriminative model will outperform a generative one. However, if data is sparse, as in the case of programming by demonstration, a generative approach tends to perform better [123].

The Bayesian approach offers the benefit of being able to apply prior knowledge to the model. Prior information informs the model and allows for more successful generalization with less data. The number of demonstrations required to teach a task or concept can be used as a metric for the effectiveness of the learner. The fewer the examples required, or the shorter the demonstration session, the more quickly the robot can begin to apply that knowledge to new contexts. Furthermore, demonstration can be tiresome and costly for the teacher.

In this chapter, I assume that the demonstration consists of exchangeable examples of a single schema. Exchangeability is the condition that the joint distribution of the observations is invariant to the ordering of the data. In this case, exchangeability implies that the order in which the demonstrator provides training examples does not affect the task being demonstrated. If exchangeability of training examples is not appropriate, or if the task requires multiple types of schema to be used, then a more general hierarchical model can be used. As mentioned in Chapter 4, if the demonstration is represented using a sequence of schemas, then a hierarchical HMM can be used. In addition, there are T task variations; each variation uses different conditions for choosing references required by the schema.

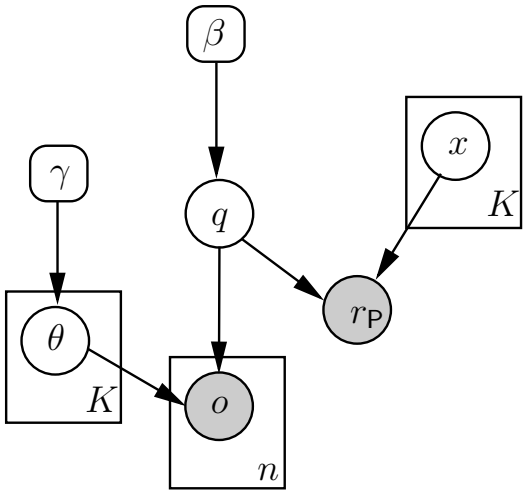

Figure 5.1. The simpler pick and place model used in this chapter. This model describes a single instantiation of pick and place. The variable q represents the latent place category for the pick and place instance with a hyperparameter β. The observed variable r_{P} is the feedback element for the Place controller. This variable is conditioned on K distinct places x. Each of the K categories has a unique distribution θ with hyperparameter γ; sampling from one of these distributions produces the n observed features in a training example.

5.1.1 A Generative Model for Pick and Place Proceduralization

From the definition of the pick and place schema in Chapter 3, there are three abstract actions, the reach-grasp behavior, B_{RG}, the reach controller R, and the place controller P, that must be proceduralized for the schema to be executed. The procedural policy must find a mapping from the observed features and task context (G, t) to elements in the control basis for feedback elements $R_S = \{r_{B_{\mathsf{RG}}}, r_{\mathsf{R}}, r_{\mathsf{P}}\}$ and effector resources $T_S = \{\tau_{B_{\mathsf{RG}}}, \tau_{\mathsf{R}}, \tau_{\mathsf{P}}\}$.

In this section, I focus on a special case of pick and place that describes a rich collections of behaviors like sorting tasks. I assume there is a single, latent feature category q associated with each "place," meaning the destination of the object after being grasped. This category abstracts the notion of the "place" destination for the controller. Given this category, the robot can proceduralize the actions in the schema.

For each set of demonstration examples, the task parameters are the same, so that the number of potential "places" for objects is fixed at K. Let $\mathcal{X}_t = \{x_1, \ldots, x_K\}, x_i \in SE(3)$, be the Cartesian position and orientation of these targets that parameterize feedback elements $\{r_i\}$ where $r_i = (\sigma_{\mathbf{x}}(x_i), \omega_i)$. In these experiments, I inform the robot of these locations prior to demonstration, but they could be inferred from demonstration using techniques similar to those used for inferring pick targets, described below.

In the PbD framework, the robot observes demonstrations and infers a model from those observations. Because a generative model describes the joint distribution of the observations and underlying variables, one can describe the stochastic process from the other causal direction. That is, the generative model describes a process for generating data observations by sampling from the joint distribution.

The generative process for a single pick and place demonstration with n feature observations is shown graphically by Figure 5.1, and outlined in the steps below:

$$q \sim \text{Multinomial}(\psi) \tag{5.5}$$

$$r_{\mathsf{P}} \,|\, q = k \sim \delta_r(k) \tag{5.6}$$

$$\tau_{\mathsf{P}} \,|\, r_{\mathsf{P}} \sim \delta_\tau(r_{\mathsf{P}}) \tag{5.7}$$

$$o_i \,|\, q = k,\, 1 \leq i \leq n \sim p(o_i \,|\, \theta_k). \tag{5.8}$$

First, in (5.5), the latent place category $q \in \{1, \ldots, K\}$ is chosen using a multinomial distribution over K elements with parameters $\psi = (\pi_1, \ldots, \pi_{K-1}), \pi_K = \sum_{i=1}^{K-1} \pi_i$. These parameters are the probabilities for each distinct value $\{1, \ldots, K\}$. For this sorting task, this distribution implies that the frequency with which a class is presented in demonstration affects the likelihood of assigning that class to a novel object. While this may be true for some tasks, in these experiments, I assume that the teacher selects exemplars based on how well they reflect the class criteria to which they belong, and not based on frequency of occurrence. Thus for the two-class case, if the demonstrator provides two examples of class 1, and one example of class 2, it does not mean that class 1 is twice as likely as class 2. Therefore, I assign ψ a uniform distribution, signifying that each class is equally likely *a priori*.

The remaining steps in the process reflect the dependence of the place location and observed features on the value of q. The feedback element for the place controller is sampled in (5.6); based on this selection, the effector resource is chosen in (5.7). The final step is to sample the feature data. Since there is a single feature type in this model, and individual feature observation o_i is sampled from the distribution

$p(o_i \mid \theta_k)$, where the parameters of the distribution θ_k are specific to the category $q = k$.

The exact form of the observation likelihood model, $p(o_i \mid \theta_k)$, will depend on the type of feature being used, and may be discrete or continuous distributions. In these experiments, visual features are computed using a background subtraction routine to segment objects in the workspace. Furthermore, these experiments use a discrete feature based on the color of the object being grasped, and a continuous feature representing the size of the segmented object in pixels.

The feedback and effector resources for the reach-grasp behavior representing the "pick" action, $r_{B_{\mathsf{RG}}}$ and $\tau_{B_{\mathsf{RG}}}$, can also be modeled using a latent category variable. However, for this type of demonstration, where the robot must predict the place category for a given object presentation, the "pick" feedback element r_{RG} is known. In Chapter 6, I discuss how the robot can select a unique pre-grasp candidate (and hence $r_{B_{\mathsf{RG}}}$) based on visual features of the object.

The process outlined in (5.5)–(5.8) describes a model for a *single* execution of pick and place, and generates a vector $\boldsymbol{o}$ of n observations and a single category assignment, q. Let $\mathcal{D} = (\boldsymbol{O}, \boldsymbol{Q})$ where $\boldsymbol{O} = \{\boldsymbol{o}_1, \ldots, \boldsymbol{o}_M\}$ and $\boldsymbol{Q} = \{q_1, \ldots, q_M\}$ be the collection of observed and latent features from a demonstration of M pick and place executions.

The purpose of the training session is to provide the robot with data it can use to infer parameters of its feature model. It uses this model to decide the most likely "place" reference of an object presented to the robot. The robot can also make a decision about whether the object should be manipulated at all, based on the objects presented in the demonstration; this is discussed in Section 5.1.3.

After demonstration, in the execution phase, when a new object is presented the robot observes features $\tilde{\boldsymbol{o}}$ and selects the most likely place category using a *maximum a posteriori* (MAP) estimate:

$$q^* \quad = \quad \underset{\tilde{q} \in K}{\operatorname{argmax}}\, p(\tilde{q} \,|\, \tilde{\boldsymbol{o}}, \mathcal{D}), \tag{5.9}$$

where, by Bayes' theorem

$$p(\tilde{q} \,|\, \tilde{\boldsymbol{o}}, \mathcal{D}) \quad \propto \quad p(\tilde{\boldsymbol{o}} \,|\, \tilde{q}, \mathcal{D})\, p(\tilde{q} \,|\, \mathcal{D}). \tag{5.10}$$

The distributions on the right side of the equation incorporate the training data. Using q^*, (5.6) and (5.7), the robot can proceduralize and execute the pick and place schema.

The procedure for computing (5.10) will differ based on the likelihood model learned from demonstration:

$$p(\tilde{\boldsymbol{o}} \,|\, \tilde{q}, \mathcal{D}) \quad = \quad \prod_{i=1}^{n} p(\tilde{o}_i \,|\, \theta_{\tilde{q}}). \tag{5.11}$$

The right hand side of (5.11) illustrates the exchangeability assumption of the feature data: the n feature observations are conditionally independent given the place category. This assumption states that the distribution is invariant to the order of the observations. Furthermore, the likelihood function depends on the type of model represented by $\theta_{\tilde{q}}$. If conjugate priors are used, then efficient closed-form solutions for (5.11) can be computed that integrate out the parameters $\theta_{\tilde{q}}$.

The second term on the right of (5.10) is a posterior distribution over place categories. If ψ is a multinomial prior over q with a symmetric Dirichlet prior with parameter β, then the standard form is:

$$p(\tilde{q} \,|\, \mathcal{D}, \beta) = p(\tilde{q} \,|\, \boldsymbol{q}, \beta) \tag{5.12}$$

$$p(\tilde{q} = l \,|\, \boldsymbol{Q}, \beta) = \mathrm{Md}(\tilde{q} = l \,|\, \boldsymbol{Q}, \beta) \tag{5.13}$$

$$= \frac{\#_l^{\boldsymbol{Q}} + \beta}{(\sum_{l'=1}^{K} \#_{l'}^{\boldsymbol{Q}}) + M\beta}, \tag{5.14}$$

where (5.12) follows from the independence relations in Figure 5.1. Because q has a multinomial distribution with a Dirichlet prior, the posterior (5.12) takes the form of the Multinomial-Dirichlet distribution shown in (5.13) [14]. This can be simplified to the fraction shown in (5.14). The symbol $\#_l^{\boldsymbol{q}}$ denotes the number of occurrences of the value l in the observation vector $\boldsymbol{q}$. As mentioned above, for many types of sorting tasks, this is a uniform distribution.

5.1.2 Experiments with a Sorting Task Using Pick and Place

As a motivating example to provide a basis for a more generalized probability model, in this section I present examples of an object sorting task that utilizes the pick and place schema. In each task instance, the concept to be learned is a two-class sorting rule for objects placed in front of the robot.

Prior to each demonstration session, the teacher selects a sorting criteria, and the demonstration consists of examples from each class. For demonstration, an object is presented to the robot, and the teleoperator uses Dexter to grasp the object and place it into one of K locations. In this section, I use a batch learning approach, where the robot is presented with every demonstrated example prior to performing the model

inference and testing the learned model. In Section 5.2 I discuss an incremental approach to this task.

After the demonstration, the robot enters the execution phase of the experiment. The user places an object in the robot's workspace, and using the feature model learned from demonstration, the robot proceduralizes and executes a pick and place schema to place the object in a location.

The objects used in these experiments are shown in Appendix B. The objects are human-scale objects that are present in a household environment. Because the robot does not model the identity of the objects, the same object presented in multiple orientations will appear as multiple task examples. The robot computes the visual features of the presented object using background subtraction for segmentation and makes a classification decision using the feature model it inferred from demonstration. In total there are 30 distinct objects in the dataset, with many objects presented in multiple orientations, giving a total of 108 different object presentations.

To illustrate the model, I perform two sets of experiments, using a different feature type in each set. In the first set of experiments, a discrete color feature is the sorting criteria. Each object's color classification is labeled by hand; the objects in each color class, along with color histograms, are shown in Appendix B, Figures B.1–B.4.

To illustrate the use of a continuous feature variable, in the second set of experiments the size of the object's segment in pixels is the classification criterion. Again, each of the 108 object presentations is labeled discretely as belonging to a Small, Medium, or Large category. This classification is shown in Appendix B, Figures B.5–B.7.

5.1.2.1 Sorting Based on Color

In the training phase of these experiments, the robot is provided with M examples of each of two color categories. The color feature is computed by associating one of $L = 64$ colors with each pixel in the object segment. The discrete set of L colors is formed by a uniform partition of HSV space. The histogram of colors is sorted in descending order, and the colors in the 20^{th} percentile and higher are included in the feature vector. The 20^{th} percentile boundary helps to reduce noise in the feature vector. Since the number of colors in the object segment varies for each object, the length of the feature vector $\tilde{\boldsymbol{o}}_m$ may vary for each object m.

Given this feature, it is necessary to define the observation likelihood model (5.8). Since each observation can take one of L different values, I use a multinomial distribution over $\tilde{o}_i$ with parameter $\psi = \theta_k$. Since this parameter also must be inferred from the data, I use a symmetric Dirichlet distribution with parameter γ as a prior over θ_k, because the Dirichlet is conjugate to the multinomial distribution. This allows the posterior over θ_k to be specified as a multinomial distribution.

As shown in Figure 5.1, there are K distinct multinomial distributions in the model; one for each place category. Given the training set $\mathcal{D} = (\boldsymbol{O}, \boldsymbol{Q})$, the inference problem is to compute the posterior for each θ_k. One can rewrite (5.10) using this model:

$$p(\tilde{q} = l \,|\, \tilde{\boldsymbol{o}}, \mathcal{D}, \gamma) \quad \propto \quad p(\tilde{\boldsymbol{o}} \,|\, \tilde{q} = l, \boldsymbol{O}, \gamma)\, p(\tilde{q} = l \,|\, \mathcal{D}), \tag{5.15}$$

where

$$p(\tilde{\boldsymbol{o}} \,|\, \tilde{q}, \boldsymbol{O}, \gamma) = \prod_{i=1}^{n} p(\tilde{o}_i \,|\, \tilde{q} = l, \boldsymbol{O}, \gamma) \tag{5.16}$$

$$= \prod_{i=1}^{n} \mathrm{Md}(\tilde{o}_i \,|\, \theta = (\boldsymbol{o}_l, \gamma)) \tag{5.17}$$

$$= \frac{\#_l^{\boldsymbol{O}} + \gamma}{(\sum_{l'=1}^{L} \#_{l'}^{\boldsymbol{O}}) + k\gamma}. \tag{5.18}$$

Using this form, the discrete feature model can handle any feature type that can be modeled by a multinomial distribution. The resulting distribution (5.18) computes the posterior probability by using a smoothed counting method, where the Dirichlet prior hyperparameter, γ, controls the amount of smoothing. The total number of parameters in this model is linear in the number of place categories.

Using the above model the robot infers the posterior over each θ_k by incorporating all $2M$ demonstrations. In the execution phase, the remainder of the objects in each category are presented to the robot, and the bin into which the objects are sorted is recorded.

Figure 5.2 shows the success rate (the number of correctly sorted objects divided by the total number of objects presented) for sorting between Red and Blue classes. There are 24 Red object presentations and 18 Blue presentations, so if $n = 1$, then one Red and one Blue presentation is demonstrated to the robot, and the other $42 - 2n = 40$ presentations are used to test the model. The objects in each color class, along with their color histograms are shown in Appendix B, Figures B.1–B.4.

Figure 5.2 shows the results of performing 10 different trials for each value of n, with error bars indicating one standard deviation. Each trial consisted of n training examples per class, and the remaining $42 - 2n$ examples were used to test the model.

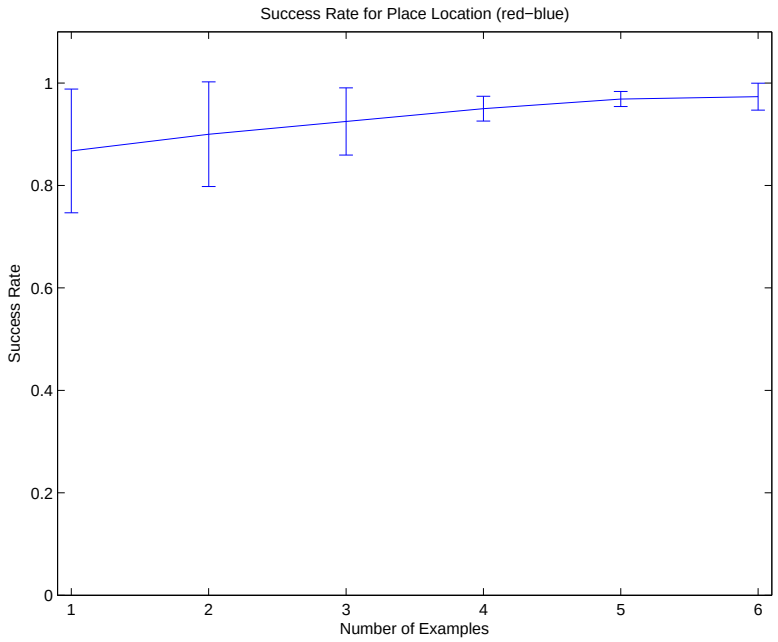

Figure 5.2. This plot shows the success rate for sorting objects across 10 trials for each number of example objects.

As one might expect, the success rate increases as the number of examples increases. However, the model is effective even with a single example of each color class. To determine the statistical significance of the sorting model, I compared it to a random sorter. I computed a p-value using a χ^2 test for each of the 10 trials. Figure 5.3 graphs the maximum and median p-value computed for each value of n. The dot–dash and dashed lines indicate 5 % and 1 % significance levels, respectively. The graph shows that in most cases the results are significant. However, for small values of n there were a few trials with less than significant sorting performance—meaning that the model did worse than randomly sorting the objects. This can be explained by the choice of feature model.

By using a multinomial distribution to model the color feature, the robot is susceptible to noisy training data. The multinomial is a frequency distribution for each color;

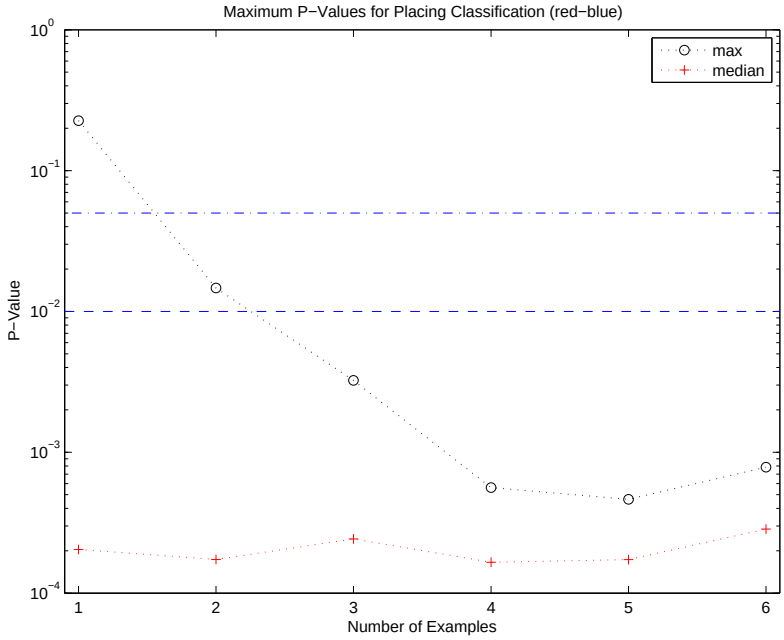

Figure 5.3. This plots the maximum p-value (out of 10 trials) per number of examples used. This p-value measures the significance of the sortable object classification.

the implication of this is that the probability assigned to color a does not affect the probability of color b, even if a and b are close together in color space. Therefore, if the training data contains colors that are not present in the test data, then classification performance will suffer. Thus, the poor trials in Figure 5.3 are the result of demonstration data that did not reflect the color values that would be seen in testing. To get an idea of how well the objects in the Red classification shared similar colors, I compiled Table 5.1.2.1. A bullet ($\bullet$) at entry (i, j) indicates the robot can successfully sort object j if object i is the only example provided. The table shows that most objects provide a sufficient range of pixel colors to identify the other objects in the class. This is evidenced by the very small median p-values shown in Figure 5.3: on average, a single object presentation from the class can be used to successfully determine the sorting criteria, however, there are outliers within the set.

	jif	jif1-1	jif1-2	jif2	oxyclean1	oxyclean1-2	oxyclean2	pink_drano1	pink_drano1-1	pink_drano1-2	pink_drano2	pink_drano2-1	pink_drano2-2	red_coffee1	red_coffee1-1	red_coffee1-2	red_football1	red_football1-1	red_pipe1	red_pipe1-1	red_pipe2	taz1	taz2	taz3
jif	●	●	●	●	●	●	●	●	●		●	●	●	●	●	●	●	●		●		●	●	●
jif1-1	●	●	●	●	●	●	●					●			●	●	●	●	●	●	●	●	●	●
jif1-2	●	●	●	●	●	●	●					●			●	●	●	●	●	●	●	●	●	●
jif2	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
oxyclean1	●	●	●	●	●	●	●					●			●	●	●	●	●	●	●	●	●	●
oxyclean1-2	●	●	●	●	●	●	●					●			●	●	●	●	●	●	●	●	●	●
oxyclean2	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
pink_drano1	●	●	●	●		●	●	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●
pink_drano1-1	●			●			●	●	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●
pink_drano1-2				●			●	●	●	●	●	●	●	●		●	●				●	●	●	●
pink_drano2	●			●			●	●	●	●	●	●	●	●		●	●		●		●	●	●	●
pink_drano2-1	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●	●
pink_drano2-2	●			●			●	●	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●
red_coffee1	●			●			●	●	●	●	●	●	●	●	●	●	●	●	●		●	●	●	●
red_coffee1-1	●	●	●	●	●	●	●	●				●			●	●	●	●	●	●	●	●	●	●
red_coffee1-2	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
red_football1	●	●	●	●	●	●	●	●	●		●	●	●	●	●	●	●	●	●	●	●	●	●	●
red_football1-1	●	●	●	●	●	●	●	●				●			●	●	●	●	●	●	●	●	●	●
red_pipe1		●	●	●	●	●	●		●		●		●	●	●	●	●	●	●	●	●	●	●	●
red_pipe1-1	●	●	●	●	●	●	●					●			●	●	●	●	●	●	●	●	●	●
red_pipe2		●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
taz1	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
taz2	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●
taz3	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●	●

Table 5.1. Red objects. This table shows how each object can be used to recognize other objects using the discrete color features. A ● indicates a recognized object; blank indicates unrecognized.

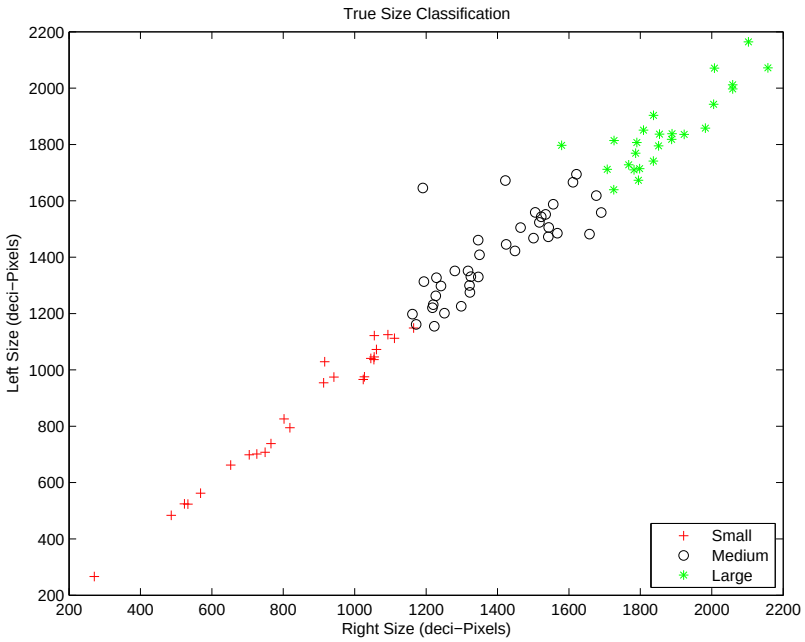

Figure 5.4. This figure shows the classification label assigned to each object presentation, based on the size of the object segment.

5.1.2.2 Sorting Based on Size

In this set of experiments, I use a continuous variable: the size of the object segment in deci-pixels (pixels/10). Each feature observation $o_i = (s_L, s_R)$ consists of stereo size measurements from Dexter's left and right cameras. The full dataset of 108 presentations was partitioned into three sets: Small (38 presentations), Medium (39), or Large (31). Appendix B, Figures B.5–B.7 show images of the presentations in each category. Figure 5.4 plots each presentation and its classification, using the left and right sizes of the object segment in deci-pixels. Deviations from diagonal in this plot are caused by perspective distortion of the object in the stereo image.

The high correlation between left and right sizes indicates that a one dimensional model captures most of the variation in the set. Therefore, I model object size using a

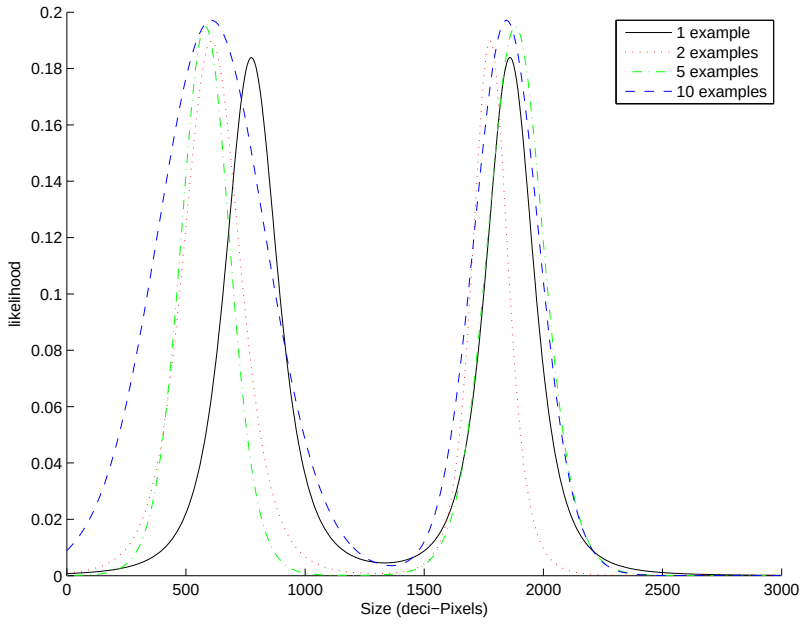

Figure 5.5. The posterior distribution over size after different numbers of training examples per category. This figure represents the same model, but each line represents the posterior updated to include a given number of training examples of each category. The distribution is bimodal, with the peak on the left accounting for Small objects, while the peak on the right describes Large objects.

one dimensional normal distribution with a Normal-Inverse-Gamma prior distribution to reduce modeling complexity. This prior is conjugate to the normal distribution; hence there exists a closed form solution for the posterior distribution of the feature given the hyperparameters. These hyperparameters reflect prior knowledge about the likely magnitude and variance of the size of objects the robot will encounter.

The demonstrator chose M presentations at random from the Small and Large classes and showed the robot the correct bin (out of K possible bins) for each object. Similar to the previous experiment, the robot learned the parameters of the pick and place feature model after receiving data from demonstration. The number of demonstrated examples in each class, M, was varied. Figure 5.5 shows an example of the posterior

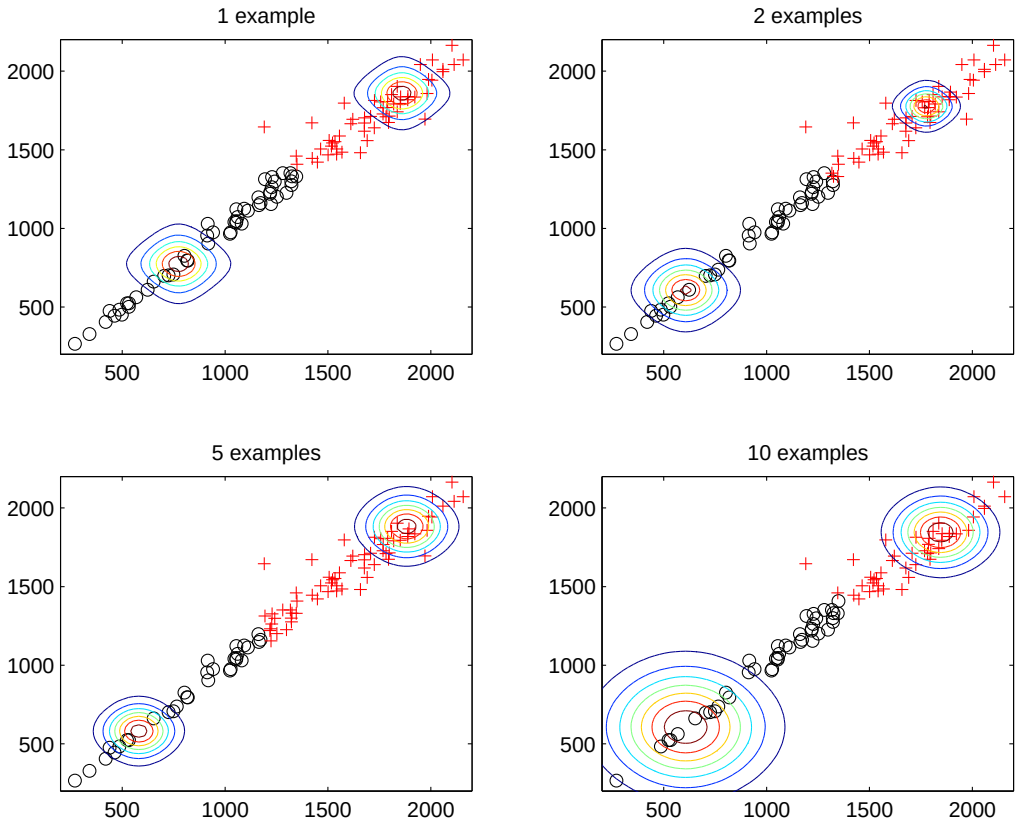

Figure 5.6. The posterior classification between Small and Large for every object instance, with varying number of training examples.

distribution that is learned for different values of M. Each line represents the same posterior, updated to reflect additional training examples. This distribution is a mixture model with the number of components equal to the number of unique place categories in the demonstration set. Each component in this posterior distribution is a Student-t distribution [14].

As M increases, the variance of each component increases, as one might expect. The mean of the Small-category component shifts to the left as the number of examples increases. This is expected because of the greater range of sizes present in the Small category, evidenced in Figure 5.4.

After learning the feature model, the remainder of the objects in the dataset were presented to the robot, and the class predicted by the model was recorded. Figure 5.6 shows the results of this classification for different values of M. The black $\circ$ represent objects that were classified as Small, and the red + represent objects classified as Large. The contour plot in each box shows the posterior distribution used to make the classification (the one dimensional likelihood was reflected about the diagonal).

The robot uses the MAP estimate of $\tilde{q}$, the predicted place category, to make the classification decision. This example shows a two-class sorting task, so the robot was forced to dichotomize every input, even those that were classified as Medium objects. Classification accuracy suffers because of this dichotomization. In Section 5.1.3, I discuss how the robot can decide whether an object should be classified using the criteria provided in the demonstration; for some objects the robot should choose to *ignore* the presentation and not execute a pick and place behavior.

The Bayesian feature model, $p(q \mid \tilde{\boldsymbol{o}}, \mathcal{D})$, uses prior distributions that reflect prior knowledge of the world. Figure 5.7 shows a contour plot of the Normal-Inverse-Gamma prior used in this experiment. The hyperparameters were chosen to broadly reflect the types of objects one expects the robot to encounter in its workspace. In this case, the typical object size is around 1300 deci-pixels, a number estimated empirically. Further, one expects that object size will have a broad variance, reflected in the prior (note the logarithmic scale used for the y-axis).

5.1.3 Prediction of Task Relevance

In the pick and place experiment, after demonstration the robot is presented with objects and must choose where to sort the object. This task can be expanded with

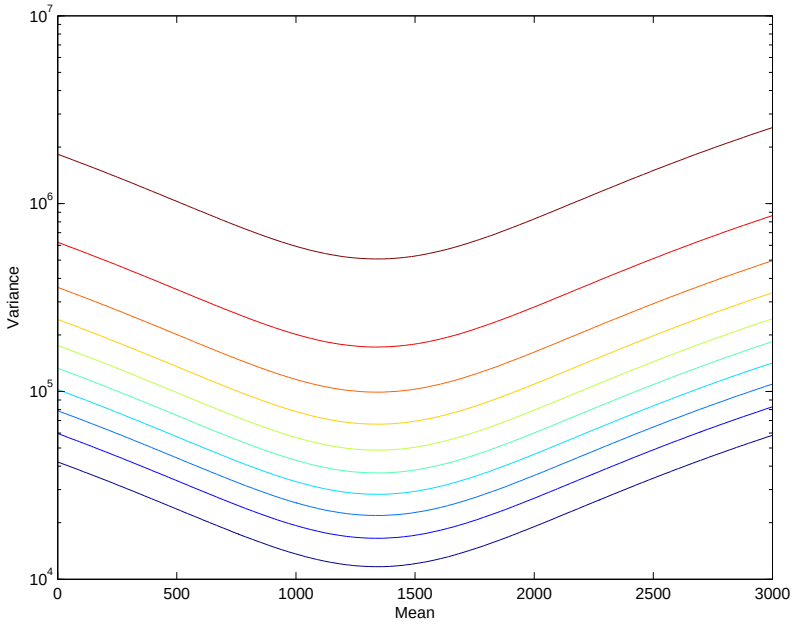

Figure 5.7. The Normal-Inverse-Gamma prior distribution; note the logarithmic scale of the Variance axis. The mean is modeled by the Normal component, and the variance by the Inverse-Gamma distribution.

an additional inference on task relevance: does a presented object belong to the set of "sortable" objects?

Let model M_0 represent a procedural feature model for non-sortable objects. The model M_1 represents the proceduralization model for the pick and place schema. The Bayes factor $\mathcal{B}_{10}$ represents the ratio of the marginal likelihood of each model given the observed features $\tilde{o}$:

$$\mathcal{B}_{10} = \frac{p_1(\tilde{o})}{p_0(\tilde{o})}. \tag{5.19}$$

If $\mathcal{B}_{10} > 1$, this indicates that M_1 explains the observed data better than M_0. When an object is presented, the robot first evaluates $\mathcal{B}_{10}$; if this indicates that M_0 is superior in explaining the observations, then the robot does not perform the pick and place

schema. If M_1 is superior, then the robot uses the proceduralization model to execute the schema.

M_0 models background objects; in the case of a color-based sorting task, consider a simple mode that assigns a uniform distribution to all colors. If the color feature can take on L discrete values, then for a single color feature $\tilde{o}_i$, M_0 predicts

$$p_0(\tilde{o}_i) = L^{-1}. \tag{5.20}$$

In the case of n color features, since all colors are equally likely, the model becomes

$$p_0(\tilde{o}_1, \ldots, \tilde{o}_n) = \prod_i p_0(\tilde{o}_i) = L^{-n}. \tag{5.21}$$

If M_1 is the proceduralization model for the schema, then the marginal likelihood of the observed data is found by integrating over model parameters Θ:

$$\begin{aligned} p_1(\tilde{p} \,|\, \Theta) &= \int p(\tilde{p} \,|\, q)\, p(q \,|\, \beta) dq & (5.22) \\ &= \sum_i \prod_j \phi_i(\tilde{p}_j)\psi(i) & (5.23) \end{aligned}$$

To illustrate this model, consider the color-based pick and place experiment. The classification problem is to determine whether the presented object should be sorted or not. Figure 5.8 shows a ROC curve for the two class sorting task between Red and Blue objects. Each curve represents a different number of training examples. In this case, because the size of the color space is relatively large, the model is able to discriminate between sortable and non-sortable colors with few examples.

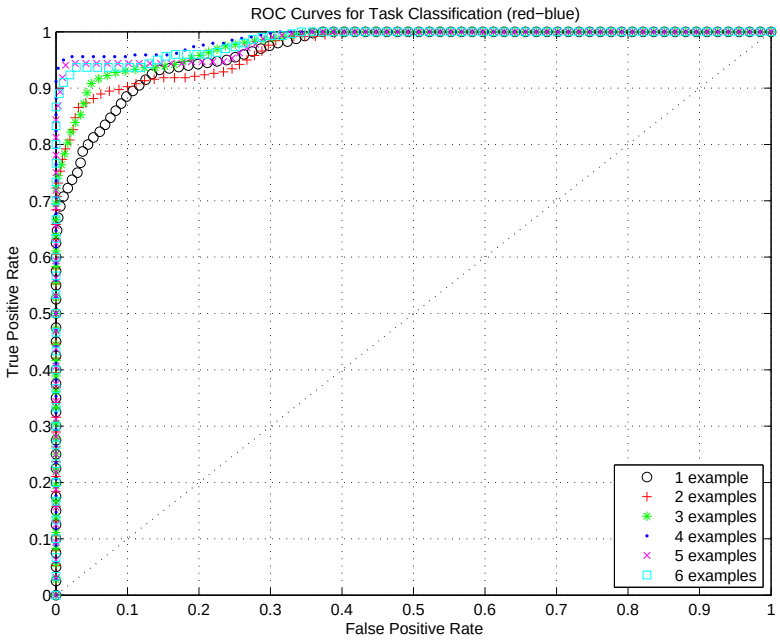

Figure 5.8. This shows the ROC curve for classifying whether an object should be sorted or not, with various number of examples.

The model for task relevance can also be extended to the case of sorting based on object size. In this case, all object presentations are partitioned into two classes: sortable and non-sortable. Figure 5.9 plots all object presentations based on segment size in the stereo image. The green $\square$ represents objects that should be sorted: they are in the Small or Large category. The magenta $+$ represents objects in the Medium category that should be ignored.

Figure 5.10 illustrates the effect of demonstration data on the decision boundary between sorting and ignoring an object presentation. In each box, the black $*$ represents the object provided by demonstration. The blue $\circ$ represents objects that the robot classifies as sortable; the red $\times$ represent objects classified as ignorable. The MAP and ML decision boundaries are shown by the blue dashed and magenta dot–dashed

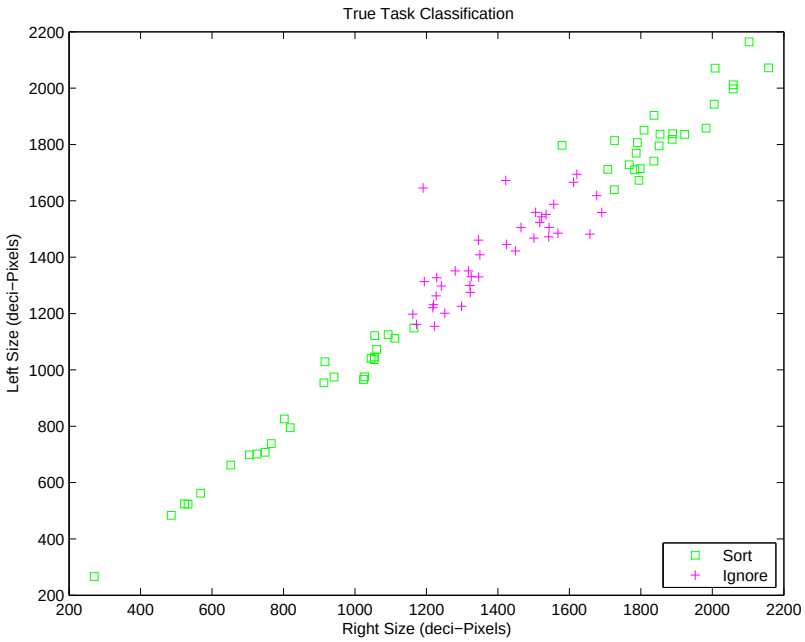

Figure 5.9. The actual task classification: each object instance is marked as sortable (□) or ignorable (○). In this task, only Medium objects were ignored.

lines, respectively. In this experiment, if either left or right size is within the MAP boundary, the object is classified as sortable.

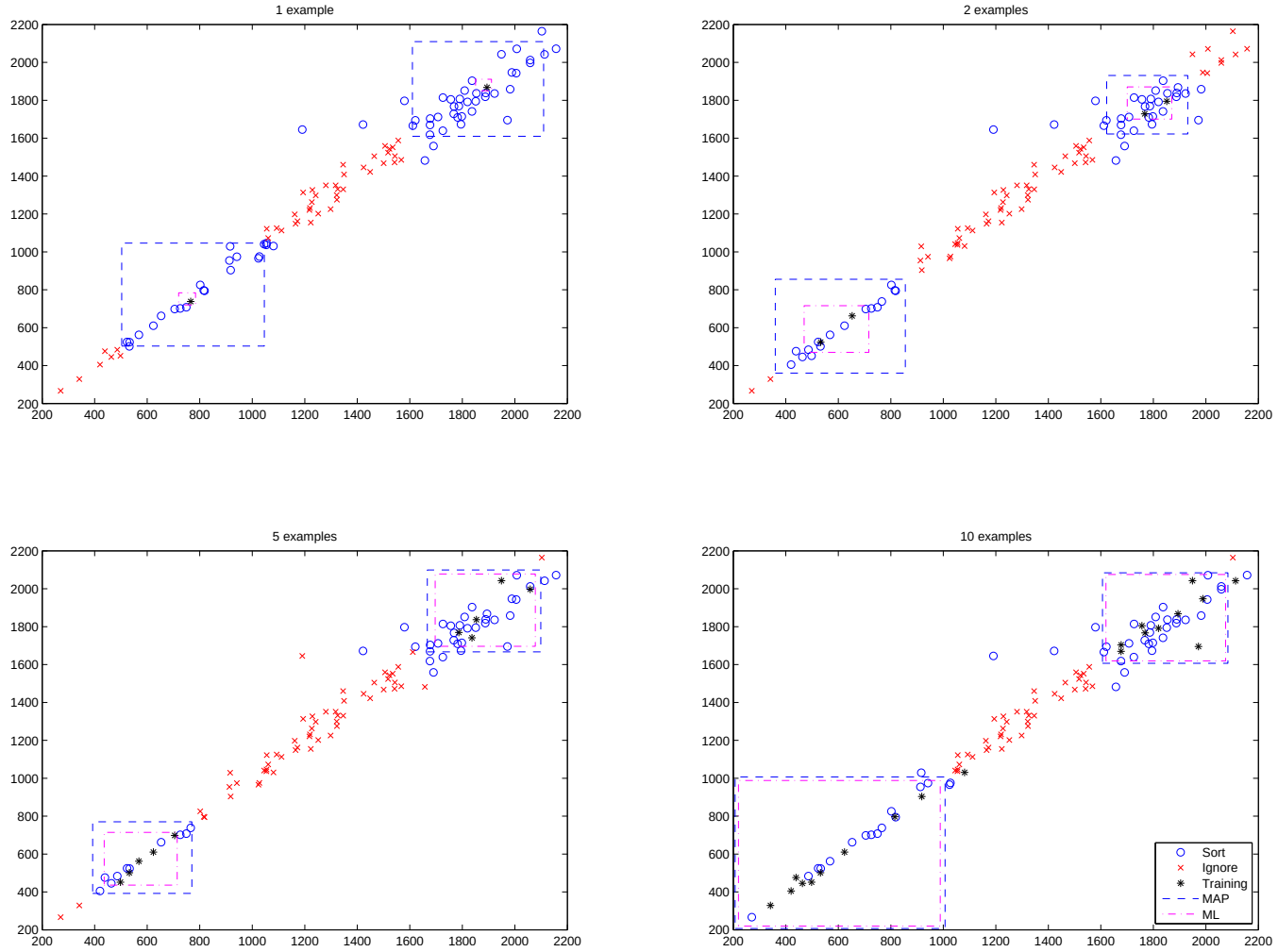

Figure 5.10. The posterior classification of task relevance. Each graph shows the learned model after a specific number of training examples, denoted by black $*$. The blue dashed line represents the learned MAP decision boundary. The magenta dashed line represents the maximum likelihood decision boundary.

The influence of the prior distribution can be seen easily by the relative size of the MAP decision boundary when the number of examples is small. As the number of examples increases, the observed data has a greater influence on the posterior, causing the MAP boundary to approach the ML boundary.

5.2 Incremental Learning

In many cases where PbD is used, it is desirable for the robot to learn incrementally. That is, as demonstrations are observed, the robot should be queried about its knowledge of the task. If the robot has not fully learned the task, then additional demonstrations can be provided. An incremental approach can help to reduce teaching load on the demonstrator, as well as provide immediate feedback on aspects of the target concept that the robot has not learned.

By using a generative model to represent the task, the robot can provide feedback to the demonstrator on what it has learned. This allows the demonstrator to decide whether the concept has been learned or what type of additional training instances are required.

An outline of an incremental demonstration session:

1. Teacher provides demonstration $\mathcal{D}_i$.
2. Robot builds model $p(X \mid \mathcal{D})$ using all demonstration data $\mathcal{D} = \bigcup_i \mathcal{D}_i$.
3. Robot generates predictive task instances by sampling from $p(\tilde{\mathcal{D}} \mid \mathcal{D})$.
4. If $\tilde{\mathcal{D}}$ reflects the task, then stop. Otherwise the teacher continues to step 1 to provide additional demonstration.

Sampling from the predictive posterior distribution $p(\tilde{\mathcal{D}} \mid \mathcal{D})$ provides a way to evaluate what the model is actually representing. It is a posterior distribution that is computed iteratively: the posterior distribution using $\{\mathcal{D}_1, \ldots, \mathcal{D}_i\}$ is used as the prior for demonstration $\mathcal{D}_{i+1}$. Furthermore, if the model makes use of exponential family components, then it suffices to retain only the sufficient statistics, and not the full dataset from each iteration [14, 80, 146].

An additional benefit of sampling from this distribution is that the teacher does not need to understand the complexities behind the model, or how the robot actually generated them. The teacher can simply evaluate whether the samples reflect the concept that has been demonstrated.

5.2.1 Multi-color Sorting Experiment

In this experiment, I perform a two-class color-based sorting task, where one class is Red and Black and the other class is Blue and White. This example illustrates how the multinomial model can represent more complicated distributions.

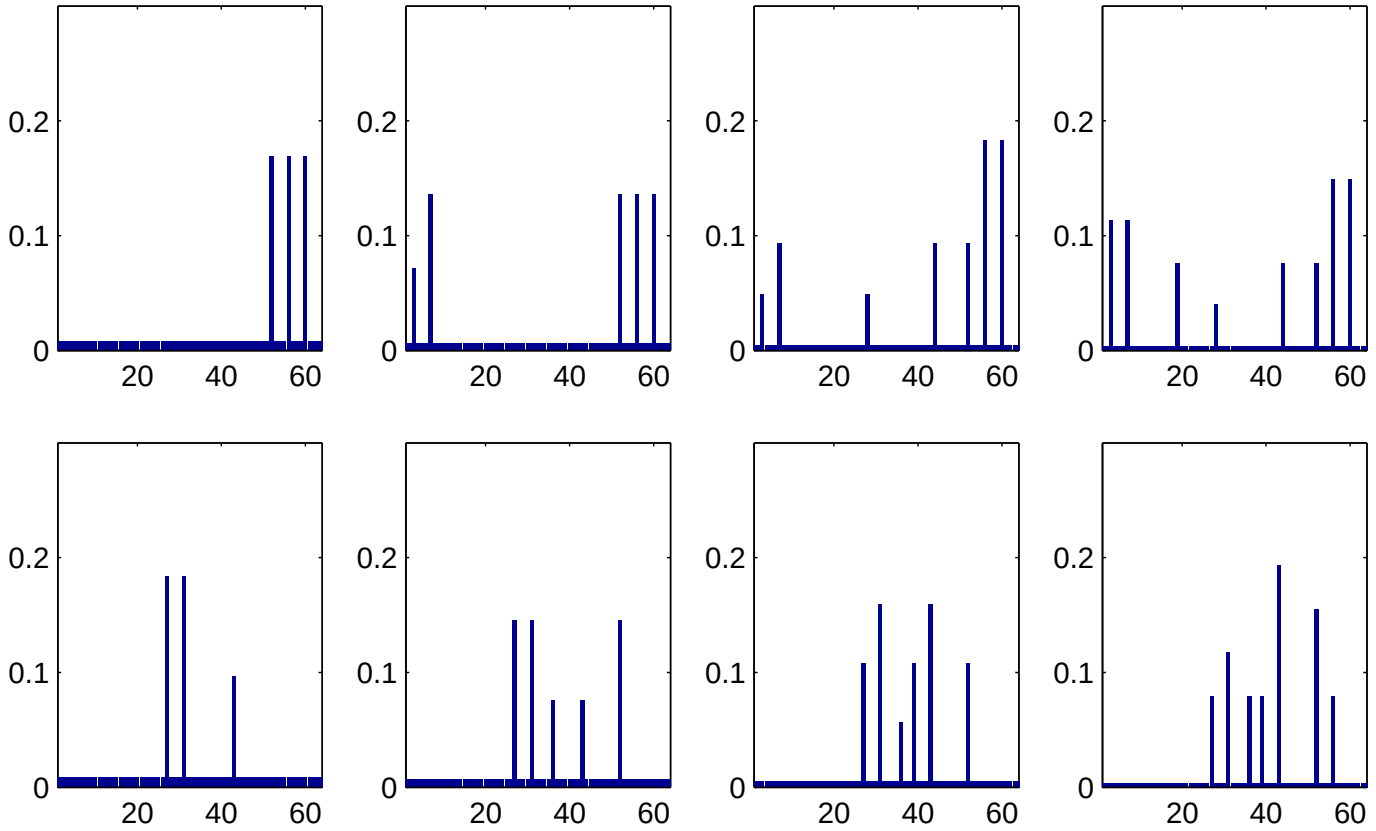

Figure 5.11. This shows the evolution of the probability mass functions; the x-axis corresponds to the set of 64 discrete colors. The top row shows the Red/Black category, and the bottom shows Blue/White. Each column, from left to right, shows the effect of an additional example provided by the demonstrator.

Figure 5.11 shows how the probability mass function for each place category changes as new examples are provided. Each row corresponds to one category: the top row is Red/Black, and the bottom is Blue/White. The columns, from left to right, correspond to each additional example provided by the demonstrator. The distribution adds mass to the color bins that are seen more often, as new data is acquired. The final shape of each probability mass function depends on the set of training examples provided. If the same training examples are provided in a different order than the one shown in Figure 5.11, the time evolution of the probability mass function will differ, but the distribution after the final example has been shown will be the same, no matter the order of the preceding examples.

As an example of how the posterior can be sampled in order to generate synthetic data, Figure 5.12 shows samples from the posterior distribution. The top and bottom figures correspond to the posterior density for the Red/Black and Blue/White categories, respectively. Each slice is a representation of the density for that category, after a particular number of examples. Each slice consists of 36 samples from the category density; these blocks are ordered within the 6×6 grid according to frequency. Since each density is defined over a 64 bin quantization of HSV space, within each of the blocks is a collection of 400 pixels uniformly sampled from that bin. Thus each slice gives an idea of which colors are more likely in that category. The slices are arranged from left to right to show the evolution of the density as new data is acquired. Each slice provides an alternative way of visualizing the shape of each place category's posterior distribution.

Figure 5.13 provides another representation for incremental changes in the posterior probability distribution. The leftmost column shows an image of each of the objects presented for training. The objects are shown from top to bottom in the order

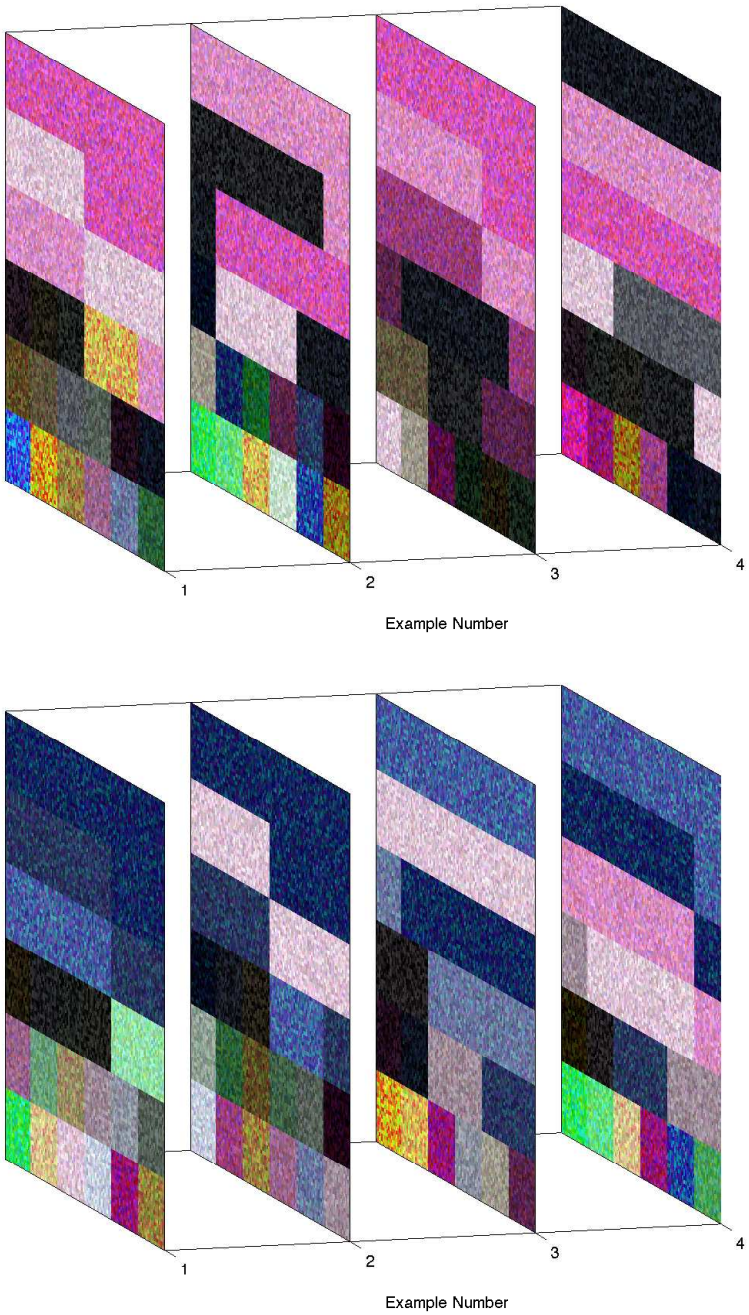

Figure 5.12. Colors generated from each pmf, as new examples are provided. Each slice represents a collection of 36 samples from the posterior distribution. Each sample is represented as a block in the slice, with the order of the blocks within the slice determined by frequency. Each sample represents one of 64 different HSV regions, and the color of the region is shown by pixels taken uniformly from the region. The slices from left to right show the effect of an additional training example. The top row corresponds to the Red/Black task, and the bottom to Blue/White. As examples are added, each slice shows the posterior distribution becomes closer to the true Red/Black or Blue/White distribution.

they are presented during demonstration. The second column shows the image after performing the color quantization process; each image is shown using a colormap with 64 entries. The remaining columns show a composite image that uses the likelihood of each quantized segment (based on the category density) to mask the original image. The purpose of this is to illustrate which portions of the image are being represented by the posterior model after integrating each example—they should correspond to mainly to the colors present in the objects shown so far. As this is done incrementally, each column shows the processed images of all objects presented up to that time. For example, the last object presented is the diagonal box shown in the bottom-left corner. The posterior distribution of the Blue and White category after incorporating this example is shown in the right-most column of the figure. The change in the posterior brought on by the additional presentation of the diagonal box can be seen by comparing the other objects in the right-most column to their neighbor to the left. The final posterior for the category will be the same no matter the order in which the objects were presented.

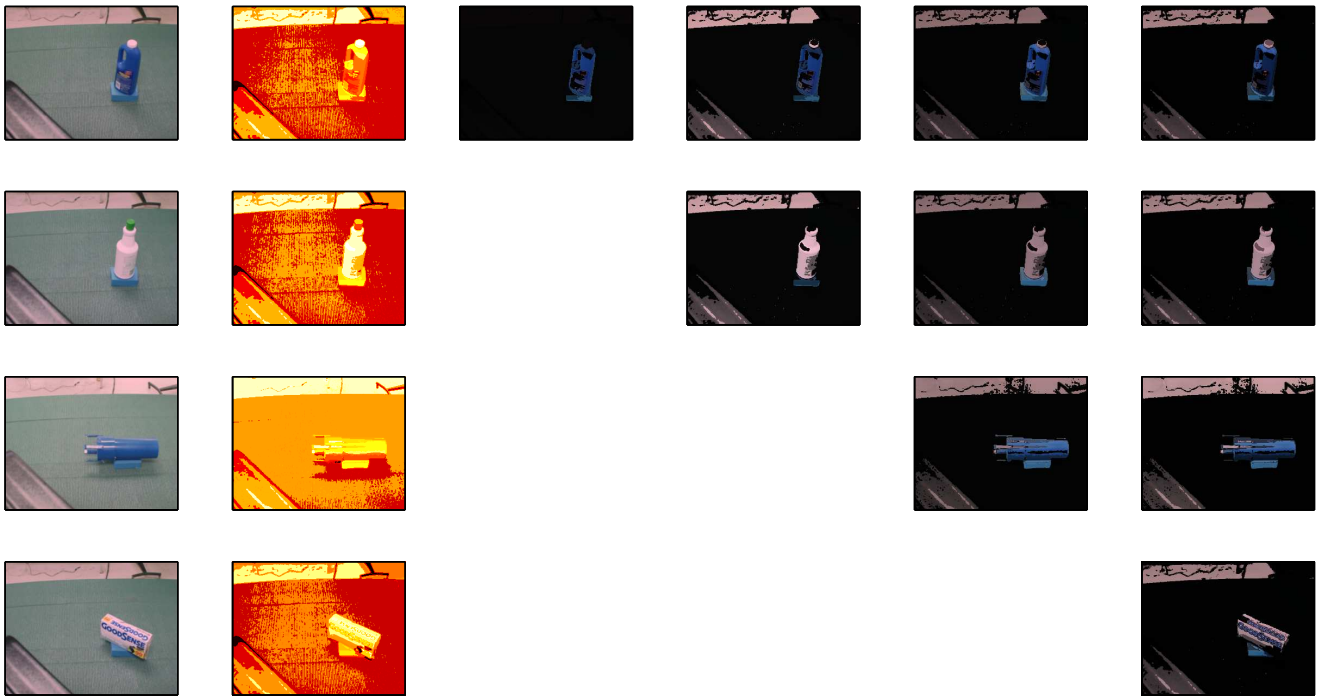

Figure 5.13. This figure shows how the model changes incrementally. The left column shows an image of the object presented for training, order from top to bottom in the order in which they were presented. The second column shows the object after color quantization into one of 64 different color values. The remaining images are constructed from the quantized image by filling in pixels based on the likelihood of the segments color according to the learned distribution. For example, the top-most row shows the likelihood of the blue bottle according to the posterior after each additional object is presented.

5.3 A General Framework for Procedural Policies from Schemas

In the preceding section I described a generative model for proceduralizing the pick and place schema, shown in Figure 5.1. In this section I describe how one can represent the schema's procedural policy

$$\psi(S, E) = p(A_S, R_S, T_S, E) \tag{5.24}$$

as a generative model using Bayes' theorem:

$$p(R_S, T_S \,|\, A_S, E) \;\; = \;\; \frac{p(A_S, E \,|\, R_S, T_S)\, p(R_S, T_S)}{p(A_S, E)}. \tag{5.25}$$

The procedural policy assigns both feedback elements and effector resources to a given schema. In the following sections I describe how this assignment can be decoupled into two mapping functions, one for effector resources, and the other for feedback elements.

5.3.1 Modeling Feedback Element Assignments

The procedural context in which a schema is executed will depend on the state of the environment and the goals that must be achieved. Let the task instance $t \in \mathbb{T}$ represent these factors that vary with each task. In order for schema S to be executed by the robot, the procedural context $\boldsymbol{r}$ must be specified for the given task instance. A probability density can be used to represent the robot's knowledge about how to assign controller references for a given schema and task:

$$p(R_S \,|\, S, t). \tag{5.26}$$

The conditional joint distribution (5.26) describes the relationship between references in a given schema for a given task instance. Although there are n components in R_S, it may not be necessary to compute n independent parameters. For a given schema, the set of feedback elements $R_S = \{r_1, \ldots, r_n\}$ can be partitioned into three distinct sets. One set describes those references that are task-independent:

$$R_S^f = \{r_i \mid \ p(r_i \,|\, r_{j \neq i}, S, t) = p(r_i \,|\, S)\}. \tag{5.27}$$

Typically, these are references for force-based controllers, where the reference value is known *a priori* and does not depend on a specific task instance. For example, the grasp controller has a zero net wrench reference, regardless of task.

Another set are those references that depend on the task instance but not on any other references in the schema:

$$R_S^t = \{r_i \mid \ p(r_i \,|\, r_{j \neq i}, S, t) = p(r_i \,|\, S, t)\}. \tag{5.28}$$

The third set are the references whose values are dependent on other references in the schema:

$$R_S^d = \{r_i \mid \ p(r_i \,|\, r_{j \neq i}, S, t) = p(r_i \,|\, \pi_S(r_i))\}, \tag{5.29}$$

where $\pi_S(r_i)$ denotes the "parent" references of r_i in this schema. The design of the schema itself determines the structure of this set. For example, if a schema has contingency behaviors, these might be dependent on the value of other references within the schema.

Thus $R_S = R_S^f \cup R_S^t \cup R_S^d$ is the complete set of references for schema S. In order to execute a schema under a specific task instance, the robot must only determine

the task-dependent references in R_S^t; the set R_S^f is known *a priori*, and R_S^d can be computed using R_S^t.

The relationship between feedback elements can also be described using a directed graph structure, where the edges represent conditional probability relationships. Figure 5.14 shows the different types of references in a schema and the conditional dependence of the effector resource on the feedback element.

For example, consider a simplified version of the pick and place schema with four abstract actions that represent the grasping action A_1, the action for reaching to the object to be grasped A_2, an action for reaching to the destination A_3, and an action to release the object A_4. In order to execute the controller, the robot must assign feedback elements to each action. The joint probability distribution over feedback elements and abstract actions given the schema and environmental context is used to select the appropriate feedback elements. Because there are conditional independence relationships between some of the elements, the full joint distribution need not be represented.

Using the distribution $p(R_S^t \mid S, t)$, the robot can compute each reference $r_i \in R_S^t$ that maximizes the likelihood $p(r_i \mid S, t)$:

$$r_i^* = \underset{r \in R}{\operatorname{argmax}}\, p(r \mid S, t). \tag{5.30}$$

5.3.2 Modeling Effector Resource Assignments

The choice of effector resource depends on the abstract action, and the feedback element. For example, a controller for reaching to Cartesian locations must be assigned a resource whose workspace contains the reference position.

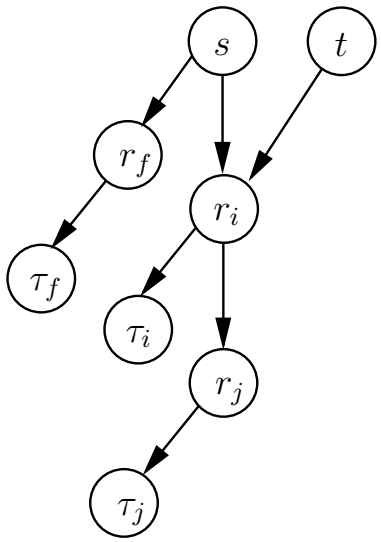

Figure 5.14. This shows the different types of feedback elements. Reference r_f is a task-independent reference, typically for force-based controllers. Reference r_i is a task dependent reference, and r_j is a reference dependent on r_i.

The relationship between feedback proceduralizations and effector resources can be described using a conditional distribution $p(T_S \mid R_S, A_S)$. Given the set of feedback references R_S, the choice of effector resource τ_i only depends on the feedback reference r_i, thus this distribution can be written as

$$p(T_S \mid A_S, R_S) = \prod_{i=1}^{n} p(\tau_i \mid a_i, r_i). \tag{5.31}$$

Given a feedback reference r_i for action a_i, each effector can be chosen separately. This assumption of conditional independence holds, since the robot will only execute one abstract action at a time (which may consists of a combination of controllers connected via nullspace projection). Therefore one can assign resources independently, knowing that the multi-objective control framework will handle any conflicts that arise. The maximum likelihood resource assignment τ_i^* can be chosen as

$$\tau_i^* = \underset{\tau \in T}{\operatorname{argmax}}\, p(\tau \mid a_i, r_i). \tag{5.32}$$

If a prior over effector resources, $p(T_S)$, is available, then it can be used with (5.32) to compute a resource allocation based on a maximum a posteriori estimate. The distribution $p(T_S \mid A_S, R_S)$ can be given to the robot *a priori*, or it may be learned from exploration [77].

5.3.3 Mapping Features to Feedback Elements

I have discussed how feedback elements are needed to execute a particular schema, and the choice of those elements may depend on the environmental context and task being completed. In this section I address how the robot computes the distribution over feedback elements. During training, the robot observes the feedback elements directly, and can use that information, along with observed environmental features to build a model used to infer feedback elements from features alone.

During demonstration, the robot has knowledge of both the elements and the features associated with the task instance. However, during the autonomous execution of the learned task, the robot must infer R based on environmental context E. This context, $E = (G, k)$, consists of the specification of the task being performed, $k \in \mathbb{T}$, along with the observable features G.

The relationship between observed features and feedback elements is captured in the covariance structure observed during demonstration.

This relationship can be summarized by the the joint distribution $p(R, G \mid k, S)$. This distribution can be represented as

$$p(R, G \mid k, S) = p(R \mid G, k, S)\, p(G \mid k, S), \tag{5.33}$$

with a *feedback element model* $p(R \mid G, k, S)$ and a *feature model* $p(G \mid k, S)$. The feature model assigns likelihood to observed features based on the selected schema and task; this likelihood is used to determine which features should be used to proceduralize the schema.

The feedback element model is used to map observed environmental features to feedback elements for abstract actions in the schema S. This distribution may be learned by the robot autonomously [76], or specified by the designer. I assume that this is part of the built-in knowledge the robot has acquired prior to observing demonstration. The feature model describes the importance of different features to the task, and represents the knowledge that the demonstrator is attempting to transfer to the robot.

The observed feature information consists of n different types of features:

$$G = \{\boldsymbol{g}_1, \ldots, \boldsymbol{g}_n\}.$$

Each component $\boldsymbol{g}_i = \{g_{i1}, \ldots, g_{im}\}$ represents a collection of m exchangeable observations of feature type i. For example, the robot may compute a collection of appearance based features, such as color, size, or shape, of objects present in the environment.

While G represents all the observable features, there may only be a subset of components of G that are relevant to the task. For example, let $G = \{\boldsymbol{g}_{\text{color}}, \boldsymbol{g}_{\text{size}}, \ldots\}$ be a collection of features including object color and size. If the task is to sort objects

based on size, then only $\boldsymbol{g}_{\text{size}}$ may be relevant. Whereas a color-based sorting task may only depend of $\boldsymbol{g}_{\text{color}}$.

If the identity of the relevant feature types are not given to the robot, then it must try and infer what they are from the demonstration. This is an instance of the *feature selection* problem studied in machine learning literature [73].

5.3.4 Creating a Feature Model

The next step is to create a graphical feature model based on the feedback elements of the schema and the task. For each $r_i \in R_S^t \cup R_S^d$, all the task-dependent feedback elements in the schema, create a latent *category* variable q_i. Each q_i is a discrete variable that takes a value in the range $1, \ldots, Q_i$. The category represents an abstract collection of feature types that are associated with the proceduralization of an abstract action.

The connectivity of the q_i variables should be the same as the members of R_S; i.e., if r_j is a child of r_i, then q_j is a child of q_i. As mentioned, each feedback element has a probabilistic model over features, called a *feature unit*. The category variable provides a way to create more complex models to describe the features correlated with a feedback reference.

To construct a feature unit for r_i, let $\boldsymbol{g}$ be the set of feature types the robot may observe, with $|\boldsymbol{g}| = N_g$. For each feature type j, let $\theta_k^j, k \in \{1, \ldots, Q_i\}$, be the parameters of a probabilistic model for that feature type for category k. This model is the likelihood of this feature type for this feature unit:

$$p(\boldsymbol{g}_j \mid \theta_k^j). \tag{5.34}$$

Thus, each category can have a distinct model for every type of feature. Furthermore, let $o_l^k, l \in \{1, \ldots, n\}$, be the set of exchangeable observations of feature type j. Let $O_i = \{o^1, \ldots, o_g^N\}$ be the full set of feature observations, and let $\Theta_i = \{\theta_1, \ldots, \theta_{Q_i}\}$ be the full set of model parameters. The feature unit $\mathcal{F}_i$ is the collection of likelihood parameters along with observed features for a feedback element i, $\mathcal{F}_i = (\Theta_i, O_i)$,

The next step is to add the feature units $\mathcal{F}_i$ to the graph. Each latent category q_i represents the conditional probability relationship between a feature unit $\mathcal{F}_i$ and the reference r_i. The connectivity structure between a node q_i and the observation variables in O_i depend on the task instance. For some tasks, it may not be necessary to connect q_i to every observable feature type o^i. For example, the set O_i represents observations of all feature types, but for some task, say, only the color feature is correlated to a reference.

Figure 5.15 shows a graphical feature model for a schema with two references, where $R^t = \{r_1\}$ and $R^d = \{r_2\}$. The black boxes on the arrows in Figure 5.15 indicate that the exact connectivity between each node q_i and its feature unit can vary depending on the task instance. This is illustrated in Figure 5.16, where there are two feature types: color (o^c) and area (o^a). The graph on the left shows a task instance for pick and place only dependent on color. The middle graph shows a task only dependent on area, and the right graph is a task instance that uses both color and area to determine place location. For this rest of this discussion, assume this structure is known and fixed.

The shaded nodes in Figure 5.15 represent observed values. Although this model is time-invariant, the features for different references are observed by the robot at different times during the execution of the schema. The robot uses the declarative

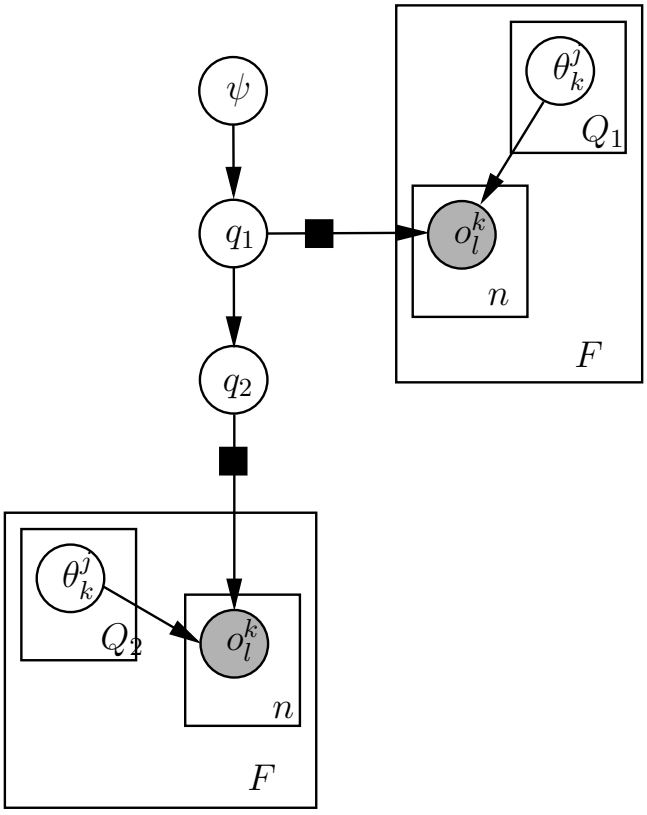

Figure 5.15. A generic feature model for a schema with two references.

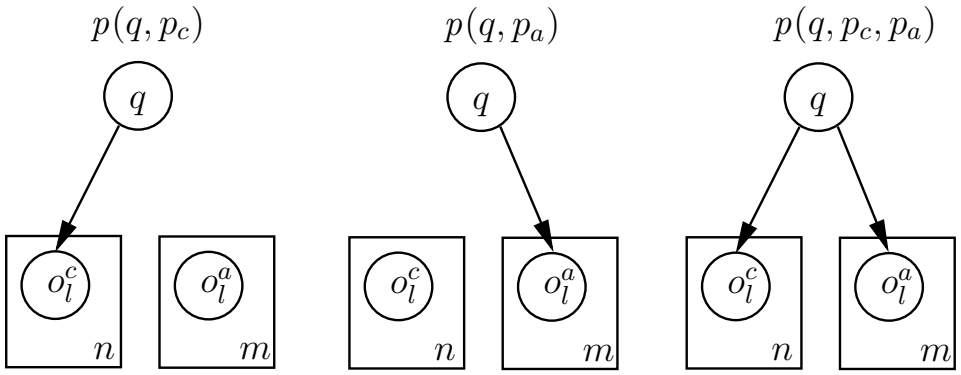

Figure 5.16. This shows the different possible relationships between q and the feature types. The lack of an arrow between nodes indicates no statistical relationship exists between the variables.

structure of the schema to determine when to observe environmental features; the teleological hypothesis suggests that declarative state transitions provide informative clues about end states. These end states encapsulate the skills being demonstrated.

5.4 Discussion

In this chapter I have outlined how a procedural policy for schemas can be learned from demonstration. The state transitions of a schema provide a natural way to focus attention on the most important parts of the demonstration; namely, where end states are observed. These goal states provide a procedural context from which the robot can learn an appropriate mapping of sensor and effector resources. The Bayesian approach I have presented here has a number of appealing aspects in the PbD domain, such as the incorporation of prior knowledge, the iterative development of the model, and the ability to generate samples from the model directly.

CHAPTER 6

INTERACTING WITH OBJECTS

The preceding chapters have outlined how a robot can learn declarative and procedural models of sensorimotor schema using demonstration data. I have focused on the pick and place schema due to its wide applicability in robot manipulation tasks. In this chapter, I describe an appearance-based model for collections of *pre-grasps*—the pose of the hand and fingers relative to the object just prior to initiating a grasping action—learned from demonstration data. The model is used to generate pre-grasps for novel objects based on visual appearance.

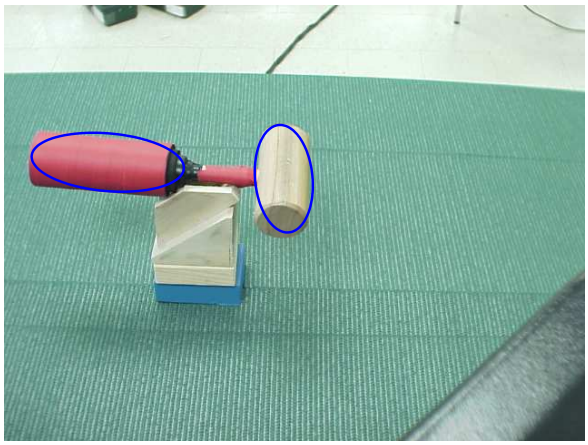

Figure 6.1. This figure shows how the mallet can be segmented into multiple segments by construction, where each segment can be used to generate grasp positions independently.

Considers grasps of the handle of the mallet presented to the robot, shown in Figure 6.1. A single pre-grasp describes the pose of the hand relative to the mallet prior to initiating a grasp. Although grasping is inherently stochastic, by using a robust grasp controller, such as the one described in Chapter 3, the robot can successfully grasp the handle with high probability from a continuous set of pre-grasps. These pre-grasps are functionally equivalent, in that each results in a grasp of the mallet's handle. As the pre-grasps move away from this set, the probability of a successful grasp decreases. It is natural to model the space of these pre-grasps using a probability distribution over relative hand and object pose. This distribution is a *grasp prototype*: a distribution over pre-grasps that initiate functionally equivalent grasps. For example, the longitudinal area over the handle of the mallet from which the robot can initiate a grasp of the handle is a single grasp prototype. An object may have multiple grasp prototypes; each ones describes a space of pre-grasps that result in a different type of grasp of the object.

The appearance of an object gives an indication to the types of grasps, and hence grasp prototypes, that it admits. Furthermore, objects with similar appearance admit similar grasps. The model presented in this chapter learns an association between the pre-grasps used by a demonstrator and the visual appearance of the object—in this case based on the first and second moments of the pixel segments, noted in Figure 6.1 by the blue ellipses. This association is a *visual grasp prototype*: a joint distribution over pre-grasps and object appearance. This distribution encodes the probability of co-occurrence of visual appearance and pre-grasps that lead to a single type of grasp. When a novel object is presented, the robot can produce candidate pre-grasps by sampling from the distribution conditioned on the object's appearance.

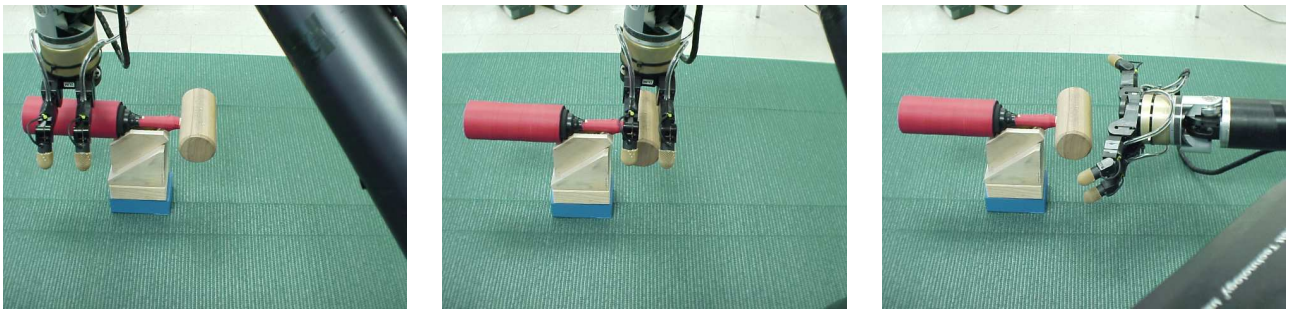

Figure 6.2. This shows different grasp locations generated by the model in Section 6.5. Each frame shows a pre-grasp associated with a distinct grasp prototype.

After observing a collection of grasp locations on a set of household objects that did not include the mallet, Figure 6.2 shows a selection of candidate pre-grasps generated by the model. Each frame represents a different visual grasp prototype of the mallet. The frame shows a single pre-grasp sampled from the distribution over relative hand and object pose encoded by the grasp prototype.

The appearance-based pre-grasp model described in this chapter is a form of statistical topic model. Topic models, traditionally used in information retrieval, model the co-occurrence of words in a text corpus [19]. Each document is represented as a collection of latent "topics," where each topic is a multinomial distribution over the words in the vocabulary. Topic models are another example of a generative model because they represent the joint probability distribution over documents and topics. The generative process demonstrates how a synthetic observation can be sampled from the model. For example, to generate a single word using a topic model, a topic is sampled from a document-specific distribution. A word is then sampled from the vocabulary given the distribution specified by the chosen topic.

$$\text{topic } t \,|\, \text{document } d \;\sim\; p(\text{topics} \,|\, d) \tag{6.1}$$

$$\text{word } w \,|\, t \;\sim\; p(\text{words} \,|\, t) \tag{6.2}$$

This process is repeated for every word in the document, producing a document that is made up of words sampled from a mixture of topics.

Applying this model to pre-grasps, each "document" is an object presented to the robot. The latent "topics" are the visual grasp prototypes from which pre-grasps are drawn. Each "word" is a single pre-grasp and co-occurring visual appearance, represented by the position and orientation of the hand with respect to the object's centroid and a visual feature.

$$\text{visual grasp prototype } v \,|\, \text{object } o \;\sim\; p(\text{visual grasp prototypes} \,|\, o) \tag{6.3}$$

$$\text{pre-grasp } p \,|\, v \;\sim\; p(\text{pre-grasps} \,|\, v) \tag{6.4}$$

Repeating the sampling process in (6.3) and (6.4) results in a set of pre-grasps for the object that may represent multiple visual grasp prototypes. These pre-grasps can be further filtered based on other run-time properties.

The set of grasp prototypes for an object is never explicitly known. They are instead a latent property of the object. A teleoperator demonstrates grasp locations to the robot, and each of these observations is modeled as a sample from a latent grasp prototype of the object. In this application, the identity of the object is also hidden; the robot must use visual features of the object to determine which prototypes are available. This changes (6.3) into

$$\text{grasp prototype } g \,|\, \text{features } f \;\sim\; p(\text{grasp prototypes} \,|\, f). \tag{6.5}$$

A tuple of parametric distributions are used to represent a visual grasp prototype, explained in Section 6.2. A probabilistic representation is used because demonstration via teleoperation is noisy, and signal quality can have a high variance, even for simple grasping actions. In the context of pre-grasp creation, generative models are preferred to discriminative ones because pre-grasps can be sampled directly from the model. Furthermore, in many cases generative models converge more quickly using less data than discriminative models, which is desirable in this domain, given the cost of acquiring demonstration data [123].

To learn the parameters of the model, all observations are clustered into A groups, where A is chosen arbitrarily. Each cluster is a collection of pre-grasps that can be thought of as being sampled from a single grasp prototype, since a grasp prototype is a joint distribution. The parameters of the distribution are inferred from the observations in the cluster using Bayesian techniques. A cluster may be made up of pre-grasps from several different objects, thus the grasp prototype it represents is in effect shared between those objects. The result of learning the model is a set of parameters for the distributions representing each of the A prototypes. The model provides prototypes that describe typical grasps seen in the training set; selecting which grasp to use to fulfill task constraints is a higher level objective subject to run-time task constraints.

The number A represents an estimate of the number of distinct prototypes present in the training set, and in this work is determined empirically. If the number is too small, then it is likely that dissimilar pre-grasps could be clustered together, resulting in prototypes that give high likelihood to unrelated pre-grasps. If A is too large, then it is likely each prototype will not generalize across objects, resulting in an over-fitted model. Although not addressed in this work, it is possible for this model to be learned

using nonparametric Bayesian techniques, outlined in Appendix A, that attempt to infer the number A along with the other model parameters from the data [146, 149].

6.1 Related Work

Statistical topic models were originally developed in the information retrieval community for modeling documents in a corpus. The author-topic model [136] and Latent Dirichlet Allocation (LDA) [19] are two examples of this approach for discovering latent topics. Griffiths and Steyvers [70] were the first to propose using Gibbs sampling to learn the parameters of the model, which is the approach used in this work.

Topic models have also been applied in the vision domain for object recognition and classification tasks [144]. The work in this dissertation is influenced in particular by the hierarchical, part-based model of Sudderth *et al.* [148]. That work describes a visual object classifier that models each object by computing a multinomial distribution over a set of globally shared "parts." Each part describes a cluster of image features. The set of parts in their model are analogous to the grasp prototypes described here. One difference is that in applying the model to new objects, I do not have a full set of features, and instead use only visual appearance to infer grasp prototypes. Unlike their model, I use the prototypes learned by the model to generate new pre-grasps on novel objects.

There has been other work on generating grasps using visual information. Saxena *et al.* [139] computes a grasping point for an object by analyzing visual features. Their model performs a logistic regression that estimates likely grasp positions in a 2D image based on the observed features. This work shares a similar motivation, to

grasp objects autonomously using vision, but the difference is that I am computing pre-grasps, and I explicitly model multiple hypotheses for each object.

Platt [127] describes a scheme for generating grasp hypotheses based on observations of the first and second moments of the foreground blob segment. I use similar visual features in this work, but I generate hypotheses from models of the training set of demonstrated grasps.

6.2 Representing Grasp Prototypes in the Model

Each visual grasp prototype is represented as a tuple of three parametric probability distributions: the visual appearance of the object, $p(b)$, the hand's position, $p(x)$, and the hand's orientation just prior to grasping, $p(w)$. I assume that the training data set consists of M different objects with N_m pre-grasp examples for object m.

6.2.1 Visual Appearance

Each grasp prototype is associated with its own probability distribution $p(b)$ over the space of visual appearance features; this is the distribution used in (6.5). This distribution relates object appearance to grasp prototypes: the robot is more likely to use a particular grasp prototype to generate hand/object postures if the object's appearance is given high likelihood by the prototype's distribution $p(b)$.

To compute the visual appearance, first the object is segmented using background subtraction. The object's centroid $\hat{O}_m \in \mathbf{R}^3$ is computed by performing a stereo triangulation on the first moment of foreground segments in the left and right image planes. The visual feature is the average second moment of the left and right blobs,

represented as a covariance matrix b_m, modeled using a two-dimensional inverse-Wishart distribution, parameterized by scale ψ with u degrees of freedom:

$$p(b_m \mid \psi, u) = \text{Inv-Wishart}_u(b_m \mid \psi). \tag{6.6}$$

This distribution is unimodal and typically used as a prior for multivariate covariance matrices [62]. After the model parameters have been learned, each prototype i has a set of parameters (ψ_i, u_i) used to compute the likelihood of an object's appearance feature according to (6.6).

6.2.2 Hand Position and Orientation

A grasp prototype describes a subspace in $SE(3)$: the set of relative hand and object poses that are associated with successful grasps of the object in a particular way. The shape of this subspace is determined by the geometries of the hand and object. This region is approximated using a parametric probability distribution, and the goal of model training is to infer the parameters of this distribution for each prototype. To approximate a grasp prototype's subspace in $SE(3)$, for computational convenience, instead of using a single distribution, the relative hand pose is decomposed into position and orientation components.

Position

The Cartesian position of a prototype is described as a three-dimensional normal distribution in object-centric space with mean μ_m and covariance Σ_m. From each training grasp point $p \in SE(3)$, the position of the hand $x_{mp} \in \mathbf{R}^3$ is modeled in a

frame centered at the centroid of the object, $\hat{O}_m$. The likelihood of that hand position for grasp prototype i is:

$$p(x_{mp} \,|\, \mu_i, \Sigma_i) = \text{Normal}(x_{mp} \,|\, \mu_i, \Sigma_i). \tag{6.7}$$

Orientation

The hand orientation of a grasp prototype is represented by using a discrete distribution over a set of Q canonical orientations. Although one can define continuous distributions over $SO(3)$ (cf. [108, 39]), empirical tests found that a discrete distribution was a simpler approach that produced satisfactory results. This technique is justified in the problem domain of grasping household objects presented on a table, as a small set of orientations can be used to describe a large number of example grasps.

Let w_{mp} be the canonical orientation that most closely matches the rotational component of the relative pose p. Each grasp prototype i defines a discrete distribution over the set of canonical orientations:

$$p(w_{mp} \,|\, \phi_i) = \phi_i(w_{mp}), \tag{6.8}$$

where ϕ_i is a Q-vector in the $(Q-1)$-simplex such that $\phi_i(q)$ is the probability of selecting orientation q.

By using a multinomial model for grasp orientation, each grasp prototype can represent multiple orientations. This is useful for dealing with the symmetries that can occur when grasping using Dexter. For example, Dexter can perform a side grasp on an object with the thumb pointing towards or away from the robot. If the training

data consists of both types of grasps, then the grasp prototype that encodes that particular side grasp will have nonzero probability of choosing both types of orientation. Note that for each grasp prototype, the probability table over orientation is built using the training data, so orientations that are more prevalent in the training set are more likely in the model.

After learning the model, each visual grasp prototype i has a tuple of parameters $\mathbb{C}_i = (\psi_i, u_i, \mu_i, \Sigma_i, \phi_i)$ that describe the component distributions over visual appearance and relative hand/object position and orientation. A pre-grasp is generated from prototype i by sampling from the distributions in (6.7) and (6.8) using the parameters $(\mu_i, \Sigma_i, \phi_i) \in \mathbb{C}_i$, described in Section 6.4.1.

6.3 The Generative Model

The previous section described how a grasp prototype is represented as a collection of three parametric distributions. The generative model presented in this section describes the statistical relationship between the parameters for a set of A grasp prototypes and the observed training data.

The graphical model is illustrated in Figure 6.3. The nodes of the graph represent random variables, with the shaded nodes denoting the observed variables. Unshaded nodes are latent variables that must be inferred from the data. Rectangles around variables denote replication, where the number of times is shown in the bottom right corner.

This model assigns one visual grasp prototype to each data observation; the values of the prototypes' distribution parameters are computed after all observations have been

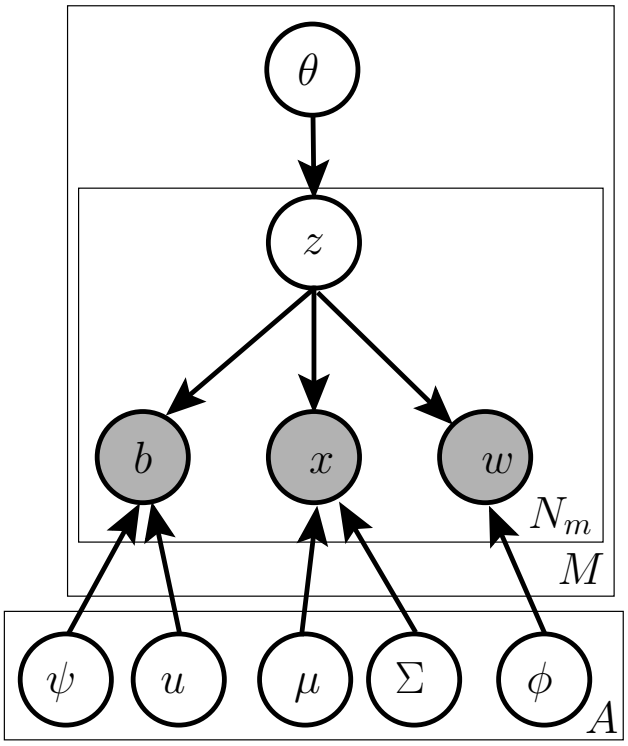

Figure 6.3. The graphical model described in Section 6.3. Circles indicate random variables, all unshaded variables are latent, shaded variables are observed. A rectangle around nodes represents replication, with the number of replications written in the bottom right corner. The edges between nodes indicates a conditional probability relationship described in the text. There are M objects and A learned visual grasp prototypes, where object m has N_m observations. θ represents the parameters of a multinomial distribution over the set of visual grasp prototypes. z is an indicator variable for one of the A prototypes. The observed variables b, x, and w are the object's visual appearance feature, and the relative hand/object position and orientation, respectively. The bottom row shows the latent distribution parameters for each of the A visual grasp prototypes: ψ and u are define the inverse-Wishart distribution for the visual feature, μ and Σ are the mean and covariance of the normal distribution describing the Cartesian position of the relative hand/object pose, and ϕ is the parameter of the multinomial distribution over hand orientation.

assigned. A Bayesian approach is used to estimate these parameters because of the ability to quantify uncertainty about their values by using suitable prior distributions.

The training data set $\mathcal{D} = (\mathbf{b}, \mathbf{x}, \mathbf{w})$ is organized into M sets of object grasp features, where set m has N_m examples. Datum i of object m is the tuple $(b_{mi}, x_{mi}, w_{mi},)$, which consists of a visual feature covariance matrix, the position, and the orientation of the hand, respectively. The variable z represents the assignment of a prototype to the observed pre-grasp. The variable θ describes a multinomial distribution over the shared set of prototypes for an object; in effect, it defines a mixture model over prototypes and describes the likelihood of an object using a particular prototype. Note that by using a multinomial distribution over the prototypes, independent samples from the model for the same object can result in a different prototype being selected.

Using this model, the generative process for example pre-grasp i is given below:

$$\theta \,|\, \alpha \sim \text{Dirichlet}(\alpha) \tag{6.9}$$

$$z_i \,|\, \theta \sim \text{Multinomial}(\theta) \tag{6.10}$$

$$b_i \,|\, z_i = j \sim \text{Inv-Wishart}_{u_j}(\psi_j) \tag{6.11}$$

$$w_i \,|\, z_i = j \sim \text{Multinomial}(\phi_j) \tag{6.12}$$

$$x_i \,|\, z_i = j \sim \text{Normal}(\mu_j, \Sigma_j). \tag{6.13}$$

Equation (6.9) samples θ from a symmetric, A-dimensional Dirichlet prior with parameter α. In (6.10), the prototype assignment z_i for this datum is sampled according to θ. The pre-grasp components b_i, w_i, and x_i are sampled according to the distributions (6.6), (6.8), and (6.7), described in Section 6.2, using the parameters of grasp prototype z_i.

The multinomial distributions θ and ϕ are given independent, symmetric Dirichlet priors with parameters α and β, respectively. The parameters of prior distributions are known as *hyperparameters*. The second moment covariance prior is inverse-Wishart with scale Ψ and u_0 degrees of freedom. The covariance matrices for grasp position, Σ, also have an inverse-Wishart prior with scale Ξ and ν degrees of freedom [62]. The grasp position mean is given a uniform prior. For notational convenience, let Ω correspond to the collection of all hyperparameters used in the model.

6.4 Parameter Estimation in the Model

The inference problem is to compute the posterior distribution of the latent variables given example pre-grasps, using Bayes' rule:

$$p(\theta, \mathbf{z}, \mathbb{C} \,|\, \mathcal{D}, \Omega) = \frac{p(\mathcal{D} \,|\, \theta, \mathbf{z}, \mathbb{C}, \Omega)\, p(\theta, \mathbf{z}, \mathbb{C} \,|\, \Omega)}{p(\mathcal{D} \,|\, \Omega)}. \tag{6.14}$$

Computing this in closed-form is intractable, but it can be estimated using Gibbs sampling. Gibbs sampling is a Markov-chain based technique that is commonly used when it is impossible to sample from a distribution directly. Instead, one samples from each dimension of the distribution conditioned on the current state of the rest of the dimensions. In this case, the distribution of interest is the posterior assignment of visual grasp prototypes to data points.

Given the data set $\mathcal{D}$, a Gibbs sampler with auxiliary parameters, as described in Neal [121], is used to estimate the grasp prototype assignments $\mathbf{z}$. These assignments are then used to provide point estimates for the other parameters θ and $\mathbb{C}$. In the following, let $\mathbf{z}_{-mi}$ denote the set of all grasp prototype assignments excluding z_{mi}, and let $\mathbf{b}_{-mi}$, $\mathbf{x}_{-mi}$, and $\mathbf{w}_{-mi}$ be defined similarly.

Using the conditional independence relationships in the graph of Figure 6.3, the posterior distribution over grasp prototype assignments can be written as

$$p(z_{mi} \mid \mathbf{z}_{-mi}, \mathcal{D}) \propto p(z_{mi} \mid \mathbf{z}_{-mi}, o_m)\, p(b_{mi} \mid \mathbf{z}, \mathbf{b}_{-mi})\, p(x_{mi} \mid \mathbf{z}, \mathbf{x}_{-mi})\, p(w_{mi} \mid \mathbf{z}, \mathbf{w}_{-mi}). \tag{6.15}$$

The likelihoods of the conditional grasp prototype assignments and hand orientation assignments are multinomials, and have been derived from standard Dirichlet integrals:

$$p(z_{mi} = j | \mathbf{z}_{-mi}, o_m = l) = \frac{n^O_{jl} + \alpha}{\sum_{j'} n^O_{jl'} + A\alpha} \tag{6.16}$$

$$p(w_{mi} = k | z_{mi} = j, \mathbf{z}_{-mi}, \mathbf{w}_{-mi}) = \frac{n^W_{kj} + \beta}{\sum_{j'} n^W_{kj'} + Q\beta}, \tag{6.17}$$

where n^O_{jl} is the number of times grasp prototype j has been assigned to object l, and A is the number of grasp prototypes shared among all objects. Likewise, n^W_{kj} is the number of times orientation feature k has been assigned to feature j, and Q is the number of canonical grasp orientations. Since the assignment of grasp prototypes to observations is a statistical process, if A is large there may be grasp prototypes that are not assigned to any observations. The expected number of grasp prototypes used is a function of the number of observations and α.

At each iteration of the sampling algorithm, given the current assignment of data points to grasp prototypes, the posterior distribution over the position of the grasp, x_{mi}, is a multivariate Student-t distribution with $(n^A_j + \nu - 2)$ degrees of freedom, where n^A_j is the total number of features assigned to grasp prototype j. This can be approximated with the following moment-matched normal distribution [62]:

$$p(x_{mi} \mid z_{mi} = j, \mathbf{z}_{-mi}, \mathbf{x}_{-mi}) \approx \mathcal{N}(x_{mi} \mid \hat{\mu}_j, \hat{\Sigma}_j), \tag{6.18}$$

where

$$\hat{\mu}_j = \frac{1}{n_j^A} \sum_{m=1}^{M} \sum_{k|z_{mk}=j} x_{mk} \tag{6.19}$$

$$\hat{\Sigma}_j = \delta_j \left(\Xi + \sum_{m=1}^{M} \sum_{k|z_{mk}=j} (x_{mk} - \hat{\mu}_j)(x_{mk} - \hat{\mu}_j)^T \right) \tag{6.20}$$

$$\delta_j = \frac{n_j^A + 1}{n_j^A (n_j^A + \nu - 4)}. \tag{6.21}$$

The conditional distribution for a visual feature covariance is given as

$$p(b_{mi} \mid z_{mi} = j, \mathbf{z}_{-mi}, \mathbf{b}_{-mi}) = \text{Inv-Wishart}_{\hat{u}_j}(\hat{\psi}_j) \tag{6.22}$$

with

$$\hat{u}_j = u_0 + n_j^A \tag{6.23}$$

$$\hat{\psi}_j = \frac{1}{n_j^A} \left(\Psi_0 + \sum_{m=1}^{M} \sum_{k|z_{mk}=j} b_{mk} \right). \tag{6.24}$$

At each iteration of the Gibbs sampler, equations (6.16) – (6.22) are used to compute (6.15). A single data point update can be computed in time $O(A)$, and each sample output by the sampler requires computing this assignment for every training data point. Thus the total time to compute a sample given a training set with M objects and N grasps per object is $O(MNA)$. The sampler must be run for a number of "burn-in" iterations before the samples can be considered independent. In this work,

on the order of two hundred "burn-in" iterations were discarded before accepting a sample.

6.4.1 Generating Pre-Grasps for New Objects

The purpose of the model is to generate candidate pre-grasps for a novel object given visual features. Let $\hat{\Theta}^{(s)}$ correspond to model parameters sampled from the posterior distribution after Gibbs iteration s. The generative process for new pre-grasps given visual feature b_t is:

$$z_t \mid b_t, \hat{\Theta}^{(s)} \sim p(z \mid b_t, \hat{\Theta}^{(s)}) \tag{6.25}$$

$$w_t \mid z_t = j, \hat{\Theta}^{(s)} \sim \text{Multinomial}(\hat{\phi}_j^{(s)}) \tag{6.26}$$

$$x_t \mid z_t = j, \hat{\Theta}^{(s)} \sim \text{Normal}(\hat{\mu}_j^{(s)}, \hat{\Sigma}_j^{(s)}). \tag{6.27}$$

With a set of samples from the posterior distribution $p(\mathbf{z} \mid \mathcal{D})$, statistics that are independent of the content of individual grasp prototypes can be computed by integrating over the full set of samples. For any single sample $\hat{\Theta}^{(s)}$, θ and $\mathbb{C}$ can be estimated using the grasp prototype assignments in $\mathbf{z}^{(s)}$ as described in Section 6.4 using (6.16) – (6.22). These correspond to predictive distributions over new grasp prototypes and hand positions conditioned on $\mathcal{D}$ and $\mathbf{z}$. Note that these estimates cannot be combined across samples, since there is no guaranteed correspondence between grasp prototypes among the set of samples.

The distribution in (6.25) can be computed as

$$p(z = i \mid b_t, \hat{\Theta}^{(s)}) \propto p(b_t \mid z = i, \hat{\Theta}^{(s)}) p(z = i \mid \hat{\Theta}^{(s)}) \approx \text{Inv-Wishart}_{\hat{u}_i^{(s)}}(\hat{\psi}_i^{(s)}), \tag{6.28}$$

where $p(z = i \mid \hat{\Theta})$ is given as uniform.

Following the generative process in (6.25)–(6.27) produces a set of candidate pre-grasps given a visual feature. In this work, all grasps are performed using a fixed configuration of the fingers in the hand, such that they form an opposing grasp. The grasp prototype representation could be amended to include a distribution over finger configuration to take into account different finger poses.

6.5 Experimental Results

To test the ability of the model to represent the grasp prototypes demonstrated in the training set and generate new pre-grasps, the model was trained using a set of household objects. Because there is no notion of orientation of the object in the model, the same object presented in a different orientation (flat, standing up, etc.) is treated as a separate object. The notation `object-N` refers to the presentation of `object` in a different orientation. There are examples in the literature of how this assumption can be relaxed by incorporating the notion of rigid body transformations into the model itself [147]. A set $\mathcal{O}$ of 31 object presentations were chosen for training and testing.

For training, $N_{train} = 19$ objects were chosen randomly from $\mathcal{O}$, and grasps were demonstrated using teleoperation. This object set is shown in Figure 6.4. Each object was presented to Dexter in the middle of the workspace, and the right arm was used to perform all grasps, as shown in Figure 4.3. While the model specifically generated pre-grasps for Dexter's right arm, by applying a known affine transformation to the pre-grasp, they can be used by the left arm. The set of canonical grasp orientations was computed using the training set, and a set of $Q = 6$ were chosen. Note that in

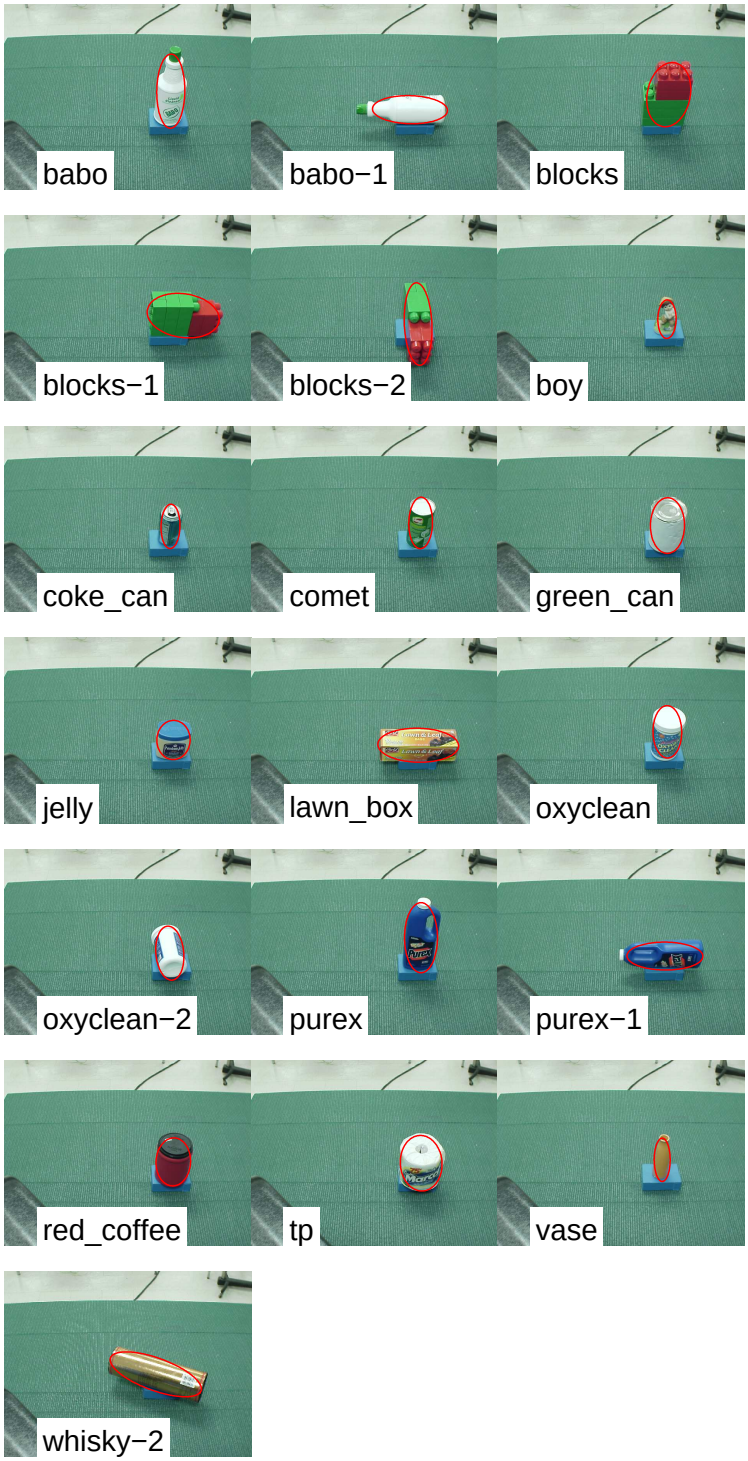

Figure 6.4. This picture shows the objects in the training set. The red oval corresponds to the covariance matrix that was used as a visual feature for grasps with the object.

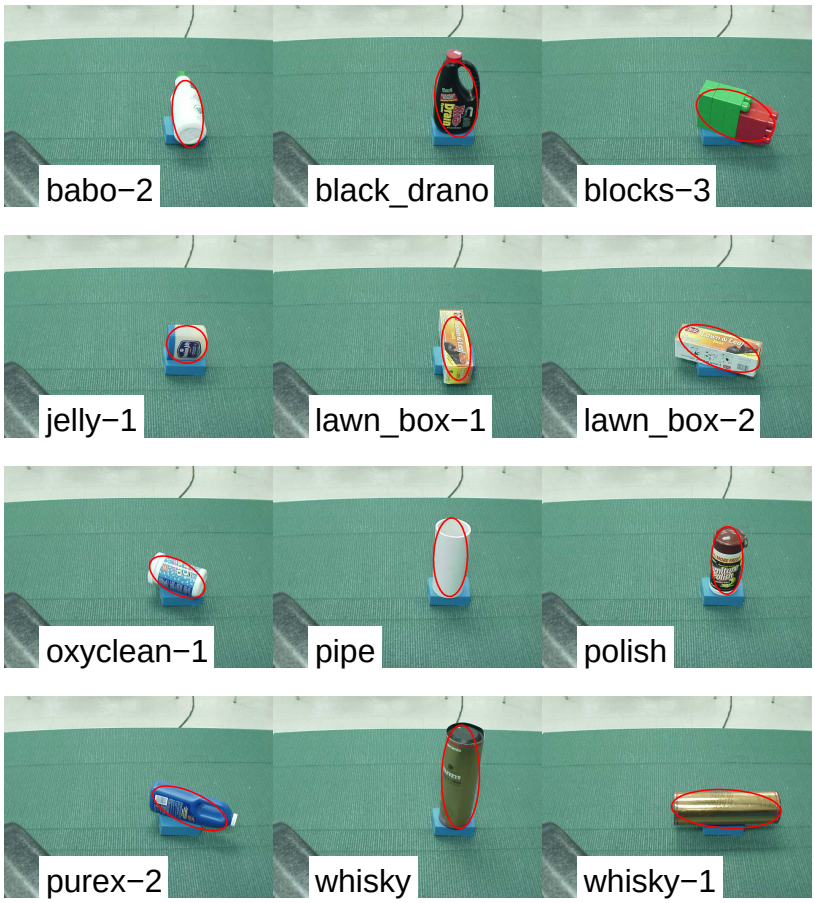

Figure 6.5. This picture shows the objects as they were presented for generating grasps.The red oval corresponds to the covariance matrix that was computed from the average second moments of the segmented blob in the left and right cameras.

these experiments, symmetric grasps were not used, that is, the demonstrator did not perform a grasp at the same location using a different hand orientation.

For learning the parameters of the model, $A = 10$ shared grasp prototypes were used. The Gibbs sampler ran for 200 iterations of burn-in, after which the next sample was stored. Using the single sample, $N_{test} = 12$ objects were presented, shown in Figure 6.5, and the model generated 6 candidate pre-grasps for each object. The robot achieved each pre-grasp configuration and then attempted to grasp the object.

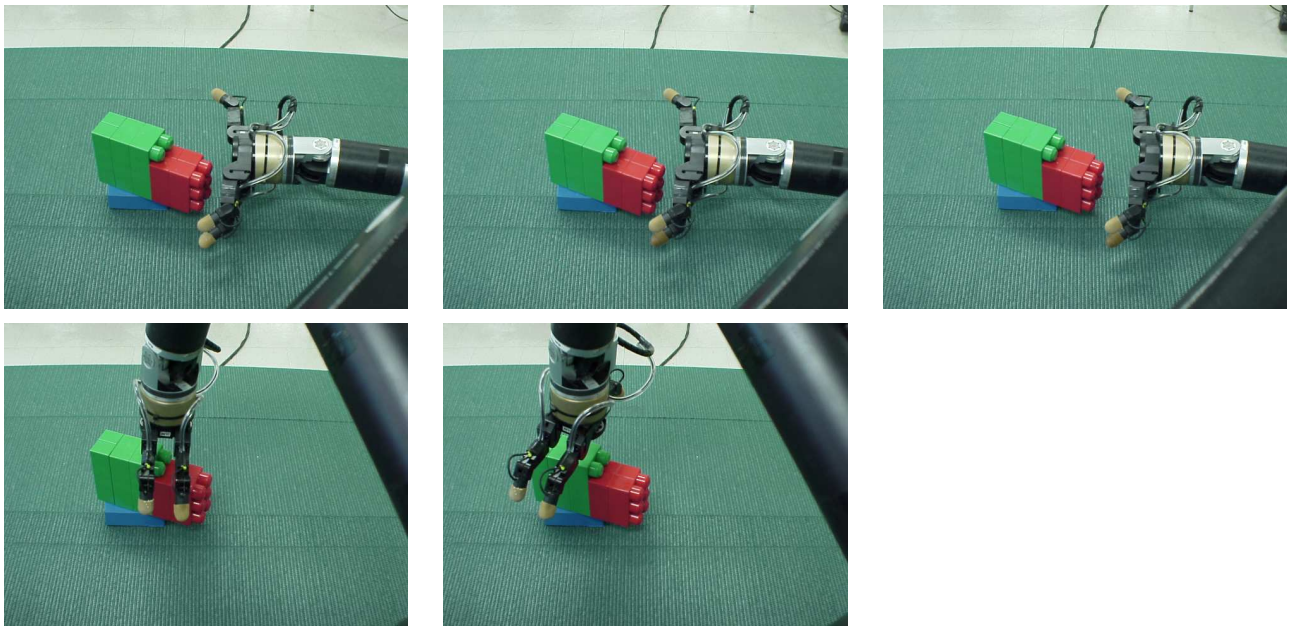

Figure 6.6. This shows different pre-grasps generated by the model in Section 6.5. The top three pre-grasps were generated by the same grasp prototype, while the bottom two pre-grasps came from two different grasp prototypes.

To perform the grasp, the robot simply flexed its fingers until a sufficient force had been applied to the object. In these experiments a grasp was judged successful if the robot was still holding onto the object after moving the hand 10 cm vertically; a sufficient distance to lift the object off the table. As an example of the types of grasps generated by the model, Figure 6.6 shows a sequence of five grasps generated for the `blocks-3` object.

6.5.1 The Naïve Model

A naïve model that also generated pre-grasps using visual features was used to analyze the performance of the appearance-based model. The naïve model performed visual processing to estimate the width and height of the object, and then generated a pre-

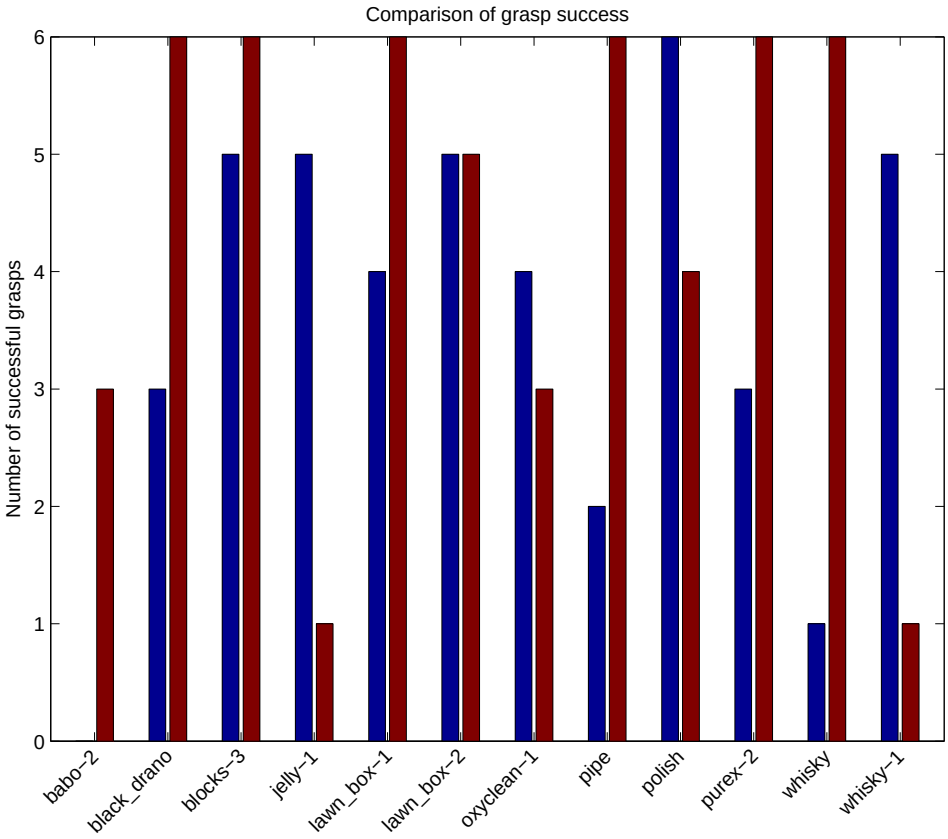

Figure 6.7. This graph shows the result of using the trained grasp model on a set of test objects. Each bar measures the number of successful grasps for the labeled object. The blue bars are for the naïve model, and the red for the visual grasp prototype model.

grasp by selecting points on a spherical hemisphere centered at the object's centroid. The radius of the hemisphere was equal to half the length of the longest dimension of the object. The orientation of the hand was chosen such that the palm was normal to a ray connecting it to the object's centroid, and a uniform random rotation about this ray was chosen. The three fingers of the hand were spread equidistant from each other. The robot then attempted to grasp the object starting from six different locations, and grasp success was judged as before.

The results of performing these grasps are shown in Figure 6.7, where blue and red bars correspond to the naïve and visual grasp prototype model, respectively. Overall, the naïve model was successful in 43 out of 72 total grasp attempts. In comparison, using the grasp prototype model, 53 out of 72 attempts were successful; a statistically significant improvement ($p < 0.01$).

In most cases, the grasp prototype model outperformed the naïve approach, including the `babo-2` object, which the naïve model was unable to grasp. However, the grasp prototype model did have difficulty with the `jelly-1` and `whisky-1` objects. In both cases, although the generated pre-grasps were located above the object with a suitable orientation, they were too high for a successful grasp. This is a result of the fact that the model is in effect summarizing the pre-grasps provided in the demonstration. For novel objects, the model finds the grasp prototype with the most similar appearance, but the hand positions suggested by that grasp prototype may not adequately fit the actual geometries of the object. To improve performance, one could incorporate a grasp controller to perform the grasp once the pre-grasp was achieved [28]. A more detailed visual appearance model would also help mitigate such failures, by reducing the chance of visually similar features not corresponding to similar, successful grasp prototypes.

Since it is based on sampling from a statistical model, the candidate pre-grasps generated by a grasp prototype may vary in quality, and in these experiments, each candidate pre-grasp was attempted regardless of its quality. However, as the amount of training data increases, the expected variance of the grasp prototype distributions will decrease, potentially improving performance. In a real-world scenario the model could be used interactively, with the teleoperator providing additional training data to improve the quality of the robot's hypotheses.

Additionally, the proposed method could be improved by performing a secondary analysis of the candidate pre-grasps. For example, incorporating additional information about the object geometry into candidate selection to choose the closest, non-colliding pre-grasp.

The success rate of the model is also affected by the number of shared grasp prototypes. In the current implementation the number A is estimated based on the number of objects presented, although one does not know *a priori* the number of shared grasp prototypes represented in the training data. If A is too small, the covariances for the position distribution of the grasp prototypes will be large, so it may require sampling a number of pre-grasps before finding one that is close enough for a successful grasp. One approach is to use a nonparametric Bayesian method that can estimate the number of grasp prototypes from the data itself [149].

Table 6.1 illustrates how visual grasp prototypes are shared among different objects, using a single sample of the posterior to show the composition of each grasp prototype. Each column corresponds to a grasp prototype, and each row denotes the training set of objects. An `x` indicates that some training grasp from this object was used to determine the parameters of the grasp prototype in that column. Columns with multiple `x`'s indicate a grasp prototype that used training examples from multiple objects. In this sample, 7 out of 10 grasp prototypes incorporate training examples from multiple objects. Once the sampler has been run for enough iterations, one can expect subsequent samples $\hat{\Theta}^{(s)}$ to contain very similar assignments. Different runs of the sampler produce similar groupings of observations, although the actual assignments to particular grasp prototypes will differ (e.g., the assignment found in grasp prototype 1 in this sample may be the assignment of grasp prototype 5 in another sample).

	Visual Grasp Prototype									
	1	2	3	4	5	6	7	8	9	10
babo		x				x				
babo-1			x							
blocks		x		x						
blocks-1	x		x							
blocks-2								x		
boy					x					
coke_can					x		x			
comet						x		x		
green_can		x		x						
jelly					x	x				
lawn_box			x							
oxyclean				x		x				
oxyclean-2								x		
purex		x		x						
purex-1									x	
red_coffee		x		x		x				
tp		x		x						
vase							x			
whisky-2										x

Table 6.1. Each `x` denotes a grasp on the object in the row that was used by the grasp prototype denoted in the column. The number of `x`'s in row i is the number of grasp prototypes used by object i. The number of `x`'s in column j is the number of objects that use that grasp prototype j.

6.6 Discussion

This chapter presents a hierarchical, statistical model for representing grasp prototypes among a collection of objects, using a form of statistical topic model. The model provides a way of summarizing the data provided by a teleoperator in a way that can be applied to new objects. The model was shown to propose candidate pre-grasps that had a statistically significant improvement over pre-grasps suggested by a baseline model.

For future work, different visual features could be used to learn grasp prototypes specific to smaller scale features of objects. For example, a set of canonical "visual words" that common visual features of the geometric structure of an object that is relevant to grasping and common across a range of objects [148, 144]. As mentioned above, the model can also be improved by incorporating rigid-body transformations into the representation of objects. Ideally, a model of grasp prototypes is learned for a canonical orientation of an object, and pre-grasps from that prototype are transformed to match the orientation of the object as it is presented. Additionally, the model can be combined with higher-level logic that selects pre-grasp candidates based on task constraints.

More Complex Objects

Using the visual feature model in this chapter, a single object can be represented with multiple second moment covariances: the model associates a grasp prototype with each covariance, and they must then be transformed into the appropriate frame. Since the model has no notion of the geometry of the object beyond centroid and second moment, secondary processing must be used to filter low-quality pre-grasps.

As an example, consider the mallet segmented into two blobs, shown in Figure 6.1. Using the model learned in the previous section, pre-grasps for each of the two blobs were generated and manually filtered to remove those that collided with the mallet. For example, the model generated pre-grasps for side grasps of the handle that would collide with the head of the mallet. Figure 6.2 shows some feasible candidate pre-grasps suggested by the model.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

In this dissertation, I have presented an architecture for humanoid robot programming by demonstration based on sensorimotor schemas. This architecture has been built using the control basis approach to control as its foundation. In the control basis, behavior is composed by associating artificial potential functions with sensor and effector resources. The control basis endows the robot with a basic level of sensorimotor behavior.

Sensorimotor schemas abstract a sequence of controller activations into procedural and declarative components, and provide the robot with a teleological, control-based perspective for observing and interpreting a demonstrator's actions. In Chapter 4, I presented a model that uses the robot's control schemas to generate a declarative representation of the demonstration. In Chapter 5, I introduced a statistical model for learning procedural policies from observation, and provided a number of experiments illustrating the potential of this approach.

I have focused on object manipulation tasks that fall under the general category of pick and place. These tasks require the robot to perform a variety of grasps, and in Chapter 6, I described an approach to pre-grasp generation based on statistical topic models. This model generalizes grasp prototypes across multiple objects, and

allows the robot to predict successful pre-grasps for novel objects. In this chapter, I recapitulate the main contributions in this dissertation, and discuss areas of future research.

7.1 Contributions

This thesis developed a computational model for robot programming by demonstration based on the assumption of a robot with a built-in set of sensorimotor behaviors known as schemas. I have decomposed the PbD problem into declarative and procedural components, and applied different statistical techniques to learn model parameters from demonstration. Furthermore, I have focused on the task domain of object manipulation with humanoid robots, and I have proposed a novel statistical model for pre-grasp generation using visual features.

This dissertation has made contributions to the state of the art in robot programming by demonstration in the following areas:

1. **Declarative Schema Inference:** In Chapter 4, I presented an architecture for declarative behavior inference from demonstration. In contrast to prior work, the robot interprets the demonstration by matching sensory observations to expected controller convergence behavior from its own set of sensorimotor schemas using dynamic statistical inference. This approach relies on the teleological aspect of closed-loop controllers: the parts of the demonstration associated with convergence events create a sequence of observations. The schema most consistent with the observations is inferred by evaluating a Hidden Markov Model.

A pick and place experiment was performed to demonstrate the efficacy of the approach, and illustrated the ability to apply knowledge observed in training to novel scenarios through the use of schemas. During this experiment, the robot observed a demonstrator perform a sorting task. The robot was able to identify its pick and place schema as the behavior most consistent with the demonstration. Using this schema, the robot was able to execute the correct sorting task. In addition, the robot used the behavioral knowledge built-in to the schema in order to execute contingencies in the pick and place schema that were never shown by the demonstrator. For example, although the demonstration was performed using Dexter's right arm, the robot executed the task using both arms, as well as transferring an object between arms, by using the declarative knowledge in the pick and place schema.

2. **Sensorimotor Schema Proceduralization:** Sensorimotor schemas are abstract representations of teleological behavior; the transitions between component controllers provide structure for inferring the proper proceduralization of the schema. In Chapter 5, I described a Bayesian approach to schema proceduralization that integrated prior and demonstrated knowledge. After the robot has observed the demonstration, the procedural model is used to apply the knowledge learned from demonstration to new environments and task instances. The proceduralization described in this chapter allows the robot to observe a a discrimination task, such as sorting, and apply the classification criteria to new environmental contexts. The chapter also outlines a general framework for developing procedural models given any sensorimotor schema.

   The approach was demonstrated experimentally using a number of sorting tasks that included classification of both discrete and continuous feature spaces. The

Bayesian approach allowed the robot to maximize the use of prior knowledge and reduce the number of demonstrations required to successfully learn a classification task.

Furthermore, the generative approach provides a novel way for the robot to communicate how much it has learned from demonstration to the instructor. By sampling synthetic proceduralizations from the posterior distribution, the instructor is able to visualize and evaluate the extent to which the robot has learned the correct policy, and refine any further demonstration in order to maximize the amount of knowledge transfer. This allows the demonstration process to be performed incrementally, with the goal of minimizing the amount of instruction needed to teach a task.

3. **Approach to Pre-Grasp Generation:** In Chapter 6, I presented a statistical model that learned a set of grasp prototypes using visual features that can be used to generate pre-grasps for objects. The model used a Bayesian approach that integrated prior knowledge and demonstration data. When presented with a novel object, the visual features of the object are associated with a learned set of grasp prototypes. Given the visual features, the appropriate grasp prototype is sampled in order to generate candidate pre-grasps that lead to successful grasps of the object. The model was able to learn grasp prototypes that were shared across multiple objects, without being explicitly told which objects should be grouped together.

   This model was tested experimentally using a set of household objects: the model parameters were learned after observing demonstration grasps on the training set of objects. After learning, objects in the test set were presented to the robot, and the pre-grasps generated by the model were evaluated. These

pre-grasps were compared to pre-grasps generated by a naïve model. The visual grasp prototype model produced candidate pre-grasps that resulted in a statistically significant improvement in grasp success compared to the naïve model.

In contrast to existing work, this model provides a way to represent grasp prototypes that are shared between objects and identifiable by visual features. The model provides a novel way of learning and representing common grasp prototypes across a collection of objects.

7.2 Looking Forward

Humanoid robots are complex machines with vast potential for improving human society. In order to achieve this, it is essential that roboticists develop powerful, natural programming methods to enable a basic competency in robot programming for everyone. Research in PbD, including the work presented in this dissertation, represent the beginnings of this exploration, with much more remaining to be developed. The work I have presented here poses additional questions and further avenues of research.

While the work presented in this dissertation has been experimentally validated on Dexter using a pick and place task domain, it is a general architecture that may be applied to others robots and tasks. I have focused on pick and place because of its wide adaptability to many different uses for object manipulation. I have used teleoperation as the method for demonstration, because of its wide use and the quality of proprioceptive feedback it provides. An immediate avenue of further research is to apply the models in this work to different experimental platforms that use different methods of demonstration. Remote observation of demonstration is the most

challenging but potentially useful method of demonstration, in that it only requires the robot be able to observe the teacher performing the task. In order to function in the remote setting, the declarative model of Chapter 4 requires that the robot have a function that can map observations to control inputs for monitors.

Although I assume that the robot has acquired a set of sensorimotor behaviors via design or through developmental methods, it is natural to learn them from demonstration. The declarative learning model could be expanded to allow for iterative refinement of existing schemas, by learning new controller convergence state transitions based on the monitors activated by demonstration. The next step would be to learn completely new schemas from demonstration.

Another avenue of further research is to expand the declarative inference model of Chapter 4. The experiments in this work demonstrated the potential of the approach, but further experimentation is required that makes use of multiple potential schemas for behavior matching. The model assumed a flat task model where each demonstration could be sufficiently explained by a single schema instance; one option for further research is to relax this assumption to allow the robot to infer sequences of schema activations from demonstration. Hierarchical extensions to HMMs provide a blueprint for how the inference model may be enhanced to handle the case of interpreting demonstrations as reusable sequences of component schemas [23].

The procedural model of Chapter 5 may be extended to other task settings that require alternative feature models. I have shown how a procedural policy using simple visual features such as color and shape can be learned from demonstration. This model may be expanded to cover a more extensive and general set of features. In the vision domain, high-dimensional image descriptors, such as SIFT [105], may provide

a basis for constructing rich, general visual features that can be applied in many different task contexts. More complex feature models are not confined to vision however; humanoids with six-axis force/torque sensing fully integrated into hand-like end effectors will be the standard, and these robots must be able to handle tasks that make use of force-domain features.

The grasp prototype model of Chapter 6 provides many areas of further research. The existing feature model can be refined to represent more complicated grasp prototypes that allow more precise and effective grasps. At a more general level, the statistical approach in the grasp prototype model can be applied to other functional relationships besides pre-grasps. For example, a welding robot should be able to recognize what types of welds are can be used with different shapes of steel, based on a collection of demonstration welds. Furthermore, the model can be improved by incorporating functional requirements into the pre-grasp sampling process. This would allow the model to learn the relationship between the grasp prototypes and the schemas that use them. When presented with novel objects, the robot can infer the valid schemas it may execute based on the grasp prototypes associated with the object.

APPENDIX A

INTRODUCTION TO BAYESIAN MODELS

A.1 Nonparametric/Semiparametric Bayesian Models

In this section I present a brief review of the Bayesian techniques and semiparametric models used to perform learning and inference in later chapters. [48]

In this thesis, I take a probabilistic approach to modeling the procedural aspects of a task. By framing the programming by demonstration paradigm in terms of setting up probability models for demonstrated tasks, the learning process that occurs post-demonstration is a statistical inference problem using the data observed during demonstration.

To illustrate the approach, consider a simple demonstrated task: single-armed reaching to a point in Dexter's workspace. Each example of the demonstration consists of the expert attempting to teleoperate Dexter's right arm to the same point, $x^\star$, in the workspace and then ending the trial. Let x_i denote the Cartesian position of the Dexter's hand at the end of trial i. Without loss of generality, assume that Dexter's hand orientation is fixed, so $x_i \in \mathbf{R}^3$.

This simple behavior of reaching to a desired location can be modeled declaratively using the reach controller, R. The controller takes a Cartesian location $r \in SE(3)$ as a reference. In this example, since hand orientation is fixed, $r \in \mathbf{R}^3$.

After demonstration, Dexter has a collection $\boldsymbol{x} = \{x_1, \ldots, x_N\}$ of N observations of the end effector's position at the end of the trial. There are a number of sources of noise in this PbD system: the measurement noise in recording the end effector locations x_i, and, more significantly in this example, the operator noise in the teleoperation signal introduced by putting a human in the loop.

To reproduce the demonstration, the problem for the robot is how to select a reference r given the examples $\boldsymbol{x}$? The probabilistic approach assumes that a statistical process with unknown parameters θ is responsible for generating each x_i. For a given task, it is reasonable to further assume that x_i is an independent and identically distributed (iid) random variable. Independent because what is demonstrated in each trial does not depend on earlier trials, and identically distributed because the same task is being demonstrated in each trial.

A further assumption about $\boldsymbol{x}$ is that the N observations are *exchangeable*: the joint probability density $p(x_1, \ldots, x_N)$ is invariant to permutations of the indices $\{i\}$ [14, 62]. In other words, the ordering of the example trials by the demonstrator does not matter. For the pick and place tasks I examine in this work, the exchangeability assumption is not a restriction.

A.1.1 Maximum Likelihood Estimates

A reasonable first choice of a model for this example is a multivariate Gaussian (normal) distribution with mean μ and a $d \times d$ covariance matrix Σ:

$$\text{Normal}(x_i \,|\, \mu, \Sigma) \;=\; (2\pi)^{-\frac{d}{2}} |\Sigma|^{-\frac{1}{2}} \exp\left\{ -\frac{1}{2}(x_i - \mu)^T \Sigma^{-1} (x_i - \mu) \right\}. \quad \text{(A.1)}$$

That x_i has a Gaussian distribution can be denoted $x_i \sim \text{Normal}(\mu, \Sigma)$. A Gaussian model, with $\theta = (\mu, \Sigma)$, for the observations $\boldsymbol{x}$ implies a unimodal representation of the target location at μ, with measurement uncertainty summarized by Σ. The mean μ is a natural choice to use for the reference r when the robot reproduces this task. The covariance matrix Σ expresses the amount of variability in the measurement of the example target locations. A demonstrator with access to Σ would have an idea about

how "well" the robot was observing the task, and how much further demonstration was required. How can the robot estimate θ given the model (A.1) and $\boldsymbol{x}$?

The *likelihood* (or *sampling distribution*) $p(\boldsymbol{x} \mid \theta)$ is the probability density of a set of observations $\boldsymbol{x}$ under a distribution with parameters θ. The *maximum likelihood* (ML) estimate finds the value $\hat{\theta}_{ML}$ that maximizes the likelihood function

$$\hat{\theta}_{ML} = \operatorname*{argmax}_{\theta} p(\boldsymbol{x} \mid \theta). \tag{A.2}$$

Because $\boldsymbol{x}$ is iid, the likelihood function can be written as the product of the likelihood of each individual observation:

$$p(\boldsymbol{x} \mid \theta) = \prod_{i=1}^{N} p(x_i \mid \theta). \tag{A.3}$$

For the multivariate Gaussian model, this likelihood function is

$$\begin{aligned} p(\boldsymbol{x} \mid \mu, \Sigma) &\propto |\Sigma|^{-\frac{n}{2}} \exp\left\{ -\frac{1}{2} \sum_{i=1}^{N} (x_i - \mu)^T \Sigma^{-1} (x_i - \mu) \right\} \\ &= |\Sigma|^{-\frac{n}{2}} \exp\left\{ -\frac{1}{2} \operatorname{tr}(\Sigma^{-1} S_\mu) \right\}, \end{aligned} \tag{A.4}$$

where $\operatorname{tr}(\cdot)$ is the trace operator, and S_μ is the sum of squares with respect to μ, $S_\mu = \sum_{i=1}^{N} (x_i - \mu)^2$.

Taking the derivative of A.4 and solving for zero, one can find the ML estimate to be the sample mean and covariance $\hat{\theta}_{ML} = (\bar{x}, S/N)$:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i \tag{A.5}$$

$$S = \sum_{i=1}^{N}(x_i - \bar{x})^2. \tag{A.6}$$

The maximum likelihood approach provides a point estimate for the mean and covariance of the model. As N increases, the maximum likelihood estimate $\hat{\theta}_{ML}$ supplies virtually all the information about θ available from the data. However, asymptotic guarantees are not as useful in programming by demonstration, where the goal is to provide the appropriate training signal in as few examples as possible.

That maximum likelihood relies solely on the observed data is also one of the drawbacks of using this approach: it ignores available prior information about θ. Consider the reaching example: if there is an obstacle in the workspace, this is information that should be included in the inference of θ.

A.1.2 Bayesian Modeling

A Bayesian analysis provides a joint probability model of the data and parameters. Conclusions about a parameter are made in terms of probability distributions, conditioned on observable data, and making use of available prior information.

The advantage of this approach is the ability to incorporate prior knowledge into the inference. In cases where the dataset is small, maximum likelihood estimates are susceptible to bias. However, consistent incorporation of prior knowledge can improve the accuracy and robustness of the Bayesian model. The Bayesian framework provides a way for the robot to apply all possible information it has acquired towards the inference problem presented by the demonstrated task.

In a Bayesian, or *generative*, model, the observations and the unobserved parameters of the underlying statistical process are modeled in a joint probability distribution

$$p(\boldsymbol{x}, \theta). \tag{A.7}$$

Using the properties of conditional probability, (A.7) can be written as

$$p(\boldsymbol{x}, \theta) \;=\; p(\boldsymbol{x} \,|\, \theta)\, p(\theta), \tag{A.8}$$

where $p(\theta)$ is known as the *prior distribution*. Often, the prior is expressed as a distribution $p(\theta \,|\, \lambda)$, where the parameters λ are known as *hyperparameters*. In a *hierarchical* analysis, the hyperparameters may also be given a prior distribution $p(\lambda)$ [14, 62]. However, for the moment, assume they are fixed with some reasonable value.

Bayes' theorem is the manipulation of (A.7) to derive the *posterior* density of θ conditioned on the observed data:

$$p(\theta \,|\, \boldsymbol{x}, \lambda) \;=\; \frac{p(\boldsymbol{x} \,|\, \theta)\, p(\theta \,|\, \lambda)}{p(\boldsymbol{x})}. \tag{A.9}$$

The denominator of (A.9) is known as the *marginal* distribution of $\boldsymbol{x}$, as it can be derived by marginalizing the joint distribution:

$$p(\boldsymbol{x}) \;=\; \int p(\boldsymbol{x}, \theta \,|\, \lambda) d\theta \tag{A.10}$$

$$\;=\; \int p(\boldsymbol{x} \,|\, \theta)\, p(\theta \,|\, \lambda) d\theta. \tag{A.11}$$

For the purpose of inferring θ, the data $\boldsymbol{x}$ remains the same, so the marginal density can be considered a constant factor. For a given set of observations, one often uses the *unnormalized posterior* density

$$p(\theta \,|\, \boldsymbol{x}, \lambda) \;\; \propto \;\; p(\boldsymbol{x} \,|\, \theta)\, p(\theta \,|\, \lambda). \tag{A.12}$$

Furthermore, because $\boldsymbol{x}$ is iid, (A.12) can be written as

$$p(\theta \,|\, \boldsymbol{x}, \lambda) \;\; \propto \;\; p(\theta \,|\, \lambda) \prod_{i=1}^{N} p(x_i \,|\, \theta). \tag{A.13}$$

Conjugate prior distributions

Bayes' theorem provides a way to compute the posterior distribution given a prior and a sampling distribution, as shown in (A.9). In some cases, by using the appropriate family of distributions to represent the model, one can compute (A.9) efficiently in closed form. A family of prior distributions $p(\theta \,|\, \lambda)$ is said to be *conjugate* to the sampling density $p(x_i \,|\, \theta)$ if, for any observation x_i and hyperparameters λ, the posterior density $p(\theta \,|\, x_i, \lambda)$ remains in that family. This can be summarized in the following relationship:

$$\begin{aligned} p(\theta \,|\, x_i, \lambda) \;\; &\propto \;\; p(x_i \,|\, \theta)\, p(\theta \,|\, \lambda) \\ &\propto \;\; p(\theta \,|\, \tilde{\lambda}), \end{aligned} \tag{A.14}$$

where $\tilde{\lambda}$ is a modified set of hyperparameters.

There exists a class of probability distributions known as the *exponential family* that are guaranteed to have the conjugate property. Members of this family include well-

known distributions such as the Bernoulli, Poisson, Gaussian, gamma, and beta distributions. Besides their use as conjugate priors, members of the exponential family have other desirable properties, such as the existence of sufficient statistics, that make them widely used in Bayesian analysis [14, 62, 146].

Using Bayesian inference, one infers a posterior distribution $p(\theta \mid \boldsymbol{x})$ over the model parameters; in contrast to the point estimate, $\hat{\theta}_{ML}$, found using maximum likelihood. Bayes' rule transfers information from the prior to the posterior distribution, and there are general properties that relate these two distributions. The law of iterated expectation can be applied to the posterior distribution of θ:

$$\begin{aligned} E[\theta] &= \int\int \theta \, p(\theta \mid \boldsymbol{x}) \, p(\boldsymbol{x}) \, d\theta \, d\boldsymbol{x} \\ &= \int E[\theta \mid \boldsymbol{x}] \, p(\boldsymbol{x}) \, d\boldsymbol{x} \\ &= E[E[\theta \mid \boldsymbol{x}]]. \end{aligned} \tag{A.15}$$

This states that the prior mean of θ is equal to the posterior mean, averaged over the distribution of the data $\boldsymbol{x}$. More interestingly, using the standard definition of variance $\mathrm{var}(u) = E[u^2] - (E[u])^2$, one can iterate expectations to derive:

$$\mathrm{var}(\theta) = E[\mathrm{var}(\theta \mid \boldsymbol{x})] + \mathrm{var}(E[\theta \mid \boldsymbol{x}]). \tag{A.16}$$

This result implies that on average, the posterior variance is less than the prior variance. The difference between the two depends on the variance of the posterior mean. To derive this result, one can work backwards:

$$
\begin{aligned}
\text{var}(\theta) &= E[\text{var}(\theta \mid \boldsymbol{x})] + \text{var}(E[\theta \mid \boldsymbol{x}]) \\
&= E[E[\theta^2 \mid \boldsymbol{x}] - (E[\theta \mid \boldsymbol{x}])^2] + E[(E[\theta \mid \boldsymbol{x}])^2] - (E[E[\theta \mid \boldsymbol{x}]])^2 \\
&= E[E[\theta^2 \mid \boldsymbol{x}]] - E[(E[\theta \mid \boldsymbol{x}])^2] + E[(E[\theta \mid \boldsymbol{x}])^2] - (E[\theta])^2 \\
&= E[\theta^2] - (E[\theta])^2.
\end{aligned}
$$

As the variation in the $\text{var}(E[\theta \mid \boldsymbol{x}])$ term increases, the uncertainty in the value of θ decreases. The posterior distribution mean represents a trade-off between the information provided by the prior and the data. This balance shifts away from the prior as the amount of data increases.

Returning to the reaching task example, given a posterior distribution of θ, there are a number of ways to select a point estimate $\hat{\theta}$ that can be used for choosing a reference for the reach controller. Some of the more common choices are the posterior mean:

$$
\bar{\theta} = E[p(\theta \mid \boldsymbol{x})], \tag{A.17}
$$

and the mode of the posterior, known as the *maximum a posteriori* estimate:

$$
\hat{\theta}_{MAP} = \underset{\theta}{\text{argmax}}\, p(\theta \mid \boldsymbol{x}). \tag{A.18}
$$

For simple parametric models with conjugate priors, (A.17) and (A.18) can be computed in closed form. However, with models using non-conjugate priors, or more complex hierarchical models, a closed form for the posterior density is typically intractable. In these cases, one can draw samples from the posterior using a sampling method to derive an empirical estimate of expectations; this is discussed in Section A.1.5.

An example using a conjugate prior

The basic model for the simple reaching example is a multivariate Gaussian distribution with d dimensions over the observed end points x_i:

$$x_i \,|\, \mu, \Sigma \;\sim\; \text{Normal}(\mu, \Sigma). \tag{A.19}$$

The Bayesian approach requires assigning prior distributions to the parameters μ and Σ. To simplify the example, assume that the covariance matrix Σ is known, thus only a prior for μ is necessary. Since the likelihood (A.19) is a member of the exponential family, a conjugate prior can be used. The conjugate prior for μ is the multivariate Gaussian distribution:

$$\mu \;\sim\; \text{Normal}(\mu_0, \Lambda_0), \tag{A.20}$$

where μ_0 and Λ_0 are the hyperparameters of the prior.

To derive a posterior density over the unknown parameter μ, combine the likelihood and prior using Bayes' theorem,

$$\begin{aligned} p(\mu \,|\, \boldsymbol{x}, \Sigma) &\propto p(\boldsymbol{x} \,|\, \mu, \Sigma)\, p(\mu) \\ &\propto \exp\left\{ -\frac{1}{2} \left[(\mu - \mu_0)^T \Lambda_0^{-1} (\mu - \mu_0) + \sum_{i=1}^{N} (x_i - \mu)^T \Sigma^{-1} (x_i - \mu) \right] \right\} \end{aligned} \tag{A.21}$$

which can be simplified to [62]:

$$p(\mu \,|\, \boldsymbol{x}, \Sigma) \;\propto\; \exp\left\{ -\frac{1}{2} (\mu - \mu_n)^T \Lambda_n^{-1} (\mu - \mu_n) \right\}, \tag{A.22}$$

where

$$\begin{aligned} \mu_n &= (\Lambda_0^{-1} + n\Sigma^{-1})^{-1}(\Lambda_0^{-1}\mu_0 + n\Sigma^{-1}\bar{x}) \\ \Lambda_n^{-1} &= \Lambda_0^{-1} + n\Sigma^{-1}. \end{aligned} \tag{A.23}$$

The distribution in (A.22) is also a multivariate Gaussian,

$$p(\mu \,|\, \boldsymbol{x}, \Sigma) = \text{Normal}(\mu \,|\, \mu_n, \Lambda_n), \tag{A.24}$$

illustrating the conjugate property of this combination of likelihood and prior.

Sequential data analysis

Another key benefit of the Bayesian approach is the natural way in which it fits into sequential data analysis problems. The posterior is updated incrementally as each observation arrives, with the posterior after $i-1$ observations acting as the prior for the i^{th}. This can be seen in the following:

$$\begin{aligned} p(\theta \,|\, x_1, \ldots, x_i, \lambda) &\propto p(x_1, \ldots, x_i \,|\, \theta)\, p(\theta \,|\, \lambda) \\ &= \prod_{j=1}^{i} p(x_j \,|\, \theta)\, p(\theta \,|\, \lambda) \\ &= p(x_i \,|\, \theta)\, p(\theta \,|\, x_1, \ldots, x_{i-1}, \lambda), \end{aligned} \tag{A.25}$$

where the second line follows from the iid assumption of the data, and the last line results from rearranging terms and applying Bayes' theorem.

A.1.3 Mixture Models

Often a single parametric distribution is insufficient for modeling a complex density. A finite *mixture model* is a convex combination of a set of kernel functions:

$$p(x \mid \theta) \;=\; \sum_{i=1}^{K} \pi_i f(x \mid \theta_i), \tag{A.26}$$

where $f(x \mid \theta_i)$ is a kernel (also referred to as a *component* or *cluster*) with parameters θ_i, and $\boldsymbol{\pi} = \{\pi_1, \ldots, \pi_K\}, \sum \pi_i = 1$, is the collection of mixture weights. Each component belongs to the same family of distributions. Mixtures using Gaussian kernels are known as *Gaussian mixture models* with parameters $\theta_i = (\mu_i, \Sigma_i)$ [62, 137, 90].

Mixture models can also be described generatively: the mixture weights $\boldsymbol{\pi}$ form a multinomial distribution over a set of latent indicator variables $z \in \{1, \ldots, K\}$. The data generating process for a single observation can be summarized as follows:

$$\begin{aligned} z_i &\sim \text{Multinomial}(\boldsymbol{\pi}) \\ x_i &\sim f(\theta_{z_i}). \end{aligned} \tag{A.27}$$

The value of indicator $z_i = j$ determines the parameter θ_j used to generate the observation. The generative approach to mixture modeling is known as *model-based clustering* [17].

An alternative form of the mixture model is to define it in terms of distributions. Let G define a measure on the parameter space of the component kernels:

$$G \;=\; \sum_{k=1}^{K} \pi_k \delta_{\theta_k} \tag{A.28}$$

where δ_{θ_k} is a Dirac distribution at θ_k. Then the generative process for obtaining a sample from such a mixture distribution is:

$$\begin{aligned} \theta_i &\sim G \\ x_i &\sim f(\theta_i) \end{aligned} \tag{A.29}$$

For a given value of K, techniques such as expectation maximization (EM) can be used to infer the parameters of the components. In practice, selecting the correct number of components is a difficult problem that has been addressed by a number of *model selection* techniques [15, 111].

Bayesian mixture models

Using EM, one can infer point estimates for the parameters of the mixture. However, in the Bayesian approach, one infers a distribution over all the parameters in the model. Additionally, priors are given to the parameters $(\boldsymbol{\theta}, \boldsymbol{\pi})$ in the mixture distribution [50]. The prior on $\boldsymbol{\theta}$ is chosen based on the form of the kernel functions. A conjugate prior can be placed on the mixture weights $\boldsymbol{\pi}$ by using a Dirichlet prior with parameters $\{\alpha_i\}$:

$$\boldsymbol{\pi} \sim \text{Dirichlet}(\alpha_1, \ldots, \alpha_K). \tag{A.30}$$

Each draw from a Dirichlet distribution is a probability mass function over K elements. If the concentration parameters are equal, $\alpha_i = \frac{\alpha}{K}$, then this is a *symmetric* Dirichlet distribution. The symmetry of this mixing prior is in accordance with the

exchangeability assumption of the underlying data: one can relabel the components and the model would remain the same.

Reaching example with a mixture model

Imagine that instead of reaching to a single point in each trial, the demonstrator reaches to one of two different locations in Dexter's workspace, depending on the presence or absence of some stimulus. The simple reaching model can be expanded by using a Gaussian mixture model $\mathcal{M}$ with $K = 2$ components to represent the data $\boldsymbol{x}$.

The generative form of this model can be represented as a graph, as shown in Figure A.1. The nodes in the graph represent variables in the model, while the edges encode the conditional probability relationships between the variables [91]. The latent parameters are the unfilled nodes, while shading denotes an observed variable, in this case each reach target x_i. Hyperparameters are represented using rounded boxes. Replication of nodes is denoted by the "plate", where the subscript indicates cardinality.

The likelihood of the model $\mathcal{M}$ given observations $\boldsymbol{x}$ is computed as:

$$
\begin{aligned}
p(\mathcal{M} \mid \boldsymbol{x}, \lambda) &\propto p(\boldsymbol{x} \mid \mathcal{M})\, p(\mathcal{M} \mid \lambda) && \text{(A.31)} \\
&= p(\boldsymbol{x} \mid \boldsymbol{\theta}, \boldsymbol{\pi})\, p(\boldsymbol{\theta}, \boldsymbol{\pi} \mid \lambda) \\
&= \prod_{i=1}^{K} p(x_i \mid z_i, \theta_i) p(z_i \mid \boldsymbol{\pi})\, p(\theta_i \mid \mu_0, \Sigma_0)\, p(\boldsymbol{\pi} \mid \alpha). && \text{(A.32)}
\end{aligned}
$$

Equation (A.31) is an application of Bayes' theorem, and the unnormalized likelihood (A.32) follows from the structure of the graph.

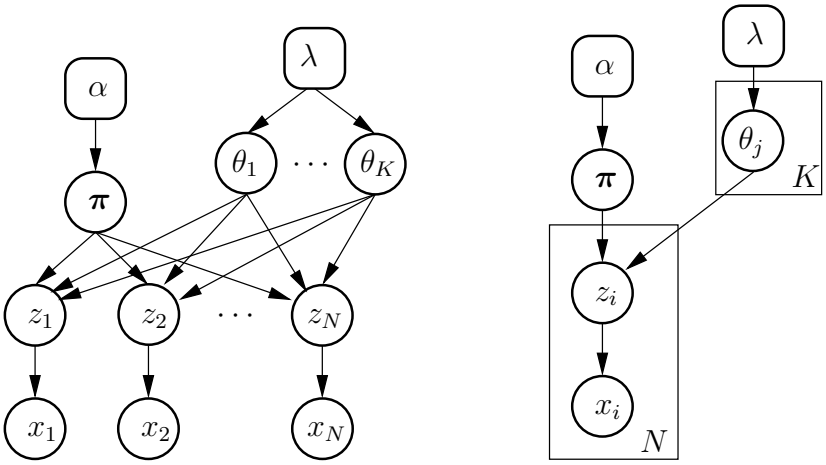

Figure A.1. The model on the left shows the Gaussian mixture model $\mathcal{M}$ in expanded form. The graph on the right uses the shorthand notation of "plates" to denote replication across variables. The α and λ variables are hyperparameters, denoted using rounded-edge boxes.

It is typically intractable to compute (A.31) in closed-form. The Monte Carlo methods discussed in Section A.1.5 can be used to efficiently draw samples from the distribution. Given enough samples, one can compute approximations to (A.31) with arbitrary accuracy.

A.1.4 Dirichlet Process

The Dirichlet process provides a way to specify a distribution over probability measures; thus it is commonly used in Bayesian nonparametric statistics as a prior over distributions [143, 18, 52, 116, 92, 68].

Consider a set of exchangeable observations, and let Θ represent a latent parameter space. A Dirichlet process is defined as follows. Let G_0 be a probability distribution on Θ, known as the *base measure*, and α a positive scalar, known as the *concentration*. A Dirichlet process is the distribution of a random probability measure G over Θ,

written $G \sim \mathrm{DP}(\alpha G_0)$, such that for any finite partition $(A_1, \ldots, A_K)$ of Θ where

$$\begin{aligned} \cup_{k=1}^{K} A_k &= \Theta \\ A_l \vee A_k &\neq \emptyset, l \neq k \end{aligned}$$

the random vector $(G(A_1), \ldots, G(A_K))$ is distributed as a Dirichlet distribution:

$$(G(A_1), \ldots, G(A_K)) \sim \mathrm{Dirichlet}(\alpha G_0(A_1), \ldots, \alpha G(A_K)). \tag{A.33}$$

Note that the parameters of the Dirichlet distribution in (A.33) are in proportion to the probability of each set A_i in the base measure. The distribution G is a continuous distribution with non-zero probability $G(A_i)$ over the set A_i.

Stick-breaking representation

An alternative representation of the Dirichlet process is known as the *stick-breaking* approach [143]. Let $\beta_k \sim \mathrm{Beta}(1, \alpha_0), k = 1, 2, \ldots$ be a sequence of Beta random variables. Using this sequence, define an infinite set of mixing proportions according to:

$$\begin{aligned} \pi_1 &= \beta_1 \\ \pi_k &= \beta_k \prod_{\ell=1}^{\ell-1} (1 - \beta_\ell), k = 2, 3, \ldots \end{aligned} \tag{A.34}$$

The process for computing the set of β_k can be seen as taking a unit-length "stick" of probability mass and breaking it at each π_i. It is clear that $\sum_{i=1}^{\infty} \pi_i = 1$. Now define the random measure $G = \sum_{k=1}^{\infty} \pi_k \delta_{\theta_k}$. It can be shown that G is also a draw from a Dirichlet process [143].

Dirichlet process mixture model

In the case of Bayesian mixture models, one is attempting to infer a distribution over the latent parameter space, Θ. In the generative view of the model, each data point x_i is a sample from a component with parameters $\theta_i \in \Theta$. The Dirichlet process provides a way to specify a prior over the space of distributions. The Dirichlet process mixture model is specified follows:

$$G \sim \mathrm{DP}(\alpha G_0) \tag{A.35}$$

$$\theta_i \sim G,\ i \in 1, \ldots, n \tag{A.36}$$

$$x_i \,|\, \theta_i \sim f(x_i \,|\, \theta_i),\ i \in 1, \ldots, n \tag{A.37}$$

Since the distribution G is in effect a discrete distribution, in (A.36) it is possible to sample the same θ for multiple data points. The location and number of distinct θ_i in the model are a function of the base measure G_0 and concentration α of the Dirichlet process.

A.1.5 Inference using Monte Carlo Methods

The result of many Bayesian analyses is a posterior distribution that one wishes to draw conclusions from, often in the form of expectations. In many cases these distributions are very complex and intractable in closed form. Markov chain Monte Carlo (MCMC) methods provide a way to find approximate solutions to these problems given sufficient computational power [107, 3, 114].

Let $p(x), x \in R^N$ represent such an intractable density. Given a set of L samples $\{x(\ell)_{\ell=1}^{L}$ from $p(x)$, one can compute the estimator $\hat{F}$ as

$$\hat{F} \;=\; \frac{1}{L}\sum_{\ell} f(x^{(\ell)}). \tag{A.38}$$

The MCMC algorithm produces samples of $p(x)$ by exploring the underlying state space using a Markov chain technique. If $x^{(0)}$ is an initial state, each subsequent state $x^{(i)}$ is generated according to the Markov process

$$x^{(i)} \;\sim\; q(x \,|\, x^{(i-1)}). \tag{A.39}$$

A transition distribution q is said to be *irreducible* if from any state there is nonzero probability of visiting all other states. An *aperiodic* chain is one free of cycles. MCMC samplers are irreducible and aperiodic Markov chains that produce the target $p(x)$ as the invariant distribution, i.e., after a number of iterations L, $x^{(L)}$ is a sample from $p(x)$ [3].

Gibbs sampling

The Gibbs sampler [61] is a specialized form of MCMC sampler that works well with state spaces where the conditional density is easily sampled from [113, 78, 3, 107].

Let $x = (x_1, \ldots, x_N)$ be the sample space decomposed into a vector of N components. At each iteration i, the idea behind the Gibbs sampler is to sample each dimension $x_j^{(i)}$ conditioned on the values of the other $N-1$ dimensions. For a single iteration, $x^{(i+1)}$ is sampled as:

$$
\begin{aligned}
x_1^{(i+1)} &\sim p(x_1 \mid x_2^{(i)}, \ldots, x_N^{(i)}) && \text{(A.40)} \\
x_2^{(i+1)} &\sim p(x_2 \mid x_1^{(i+1)}, x_3^{(i)}, \ldots, x_N^{(i)}) && \text{(A.41)} \\
&\vdots \\
x_N^{(i+1)} &\sim p(x_N \mid x_1^{(i+1)}, \ldots, x_{N-1}^{(i+1)}). && \text{(A.42)}
\end{aligned}
$$

For many Bayesian models, the conditional densities are tractable, particularly if conjugate priors are used, and one can often generate samples using closed forms. I use Gibbs sampling extensively in the later chapters of this thesis to derive approximations to posterior distributions of interest.

APPENDIX B

OBJECTS USED IN EXPERIMENTS

B.1 Objects Used in Experiments

This appendix details the objects used in the experiments of Chapter 5. Figures B.1–B.4 show each object presentation associated with the color categories used in Section 5.1.2. The color categories were assigned by hand. The inset is a normalized histogram over the 64 value HSV color space, computed over all the pixels in the object segment.

Figures B.5–B.7 show each object presentation, grouped according to its hand-labeled size category.

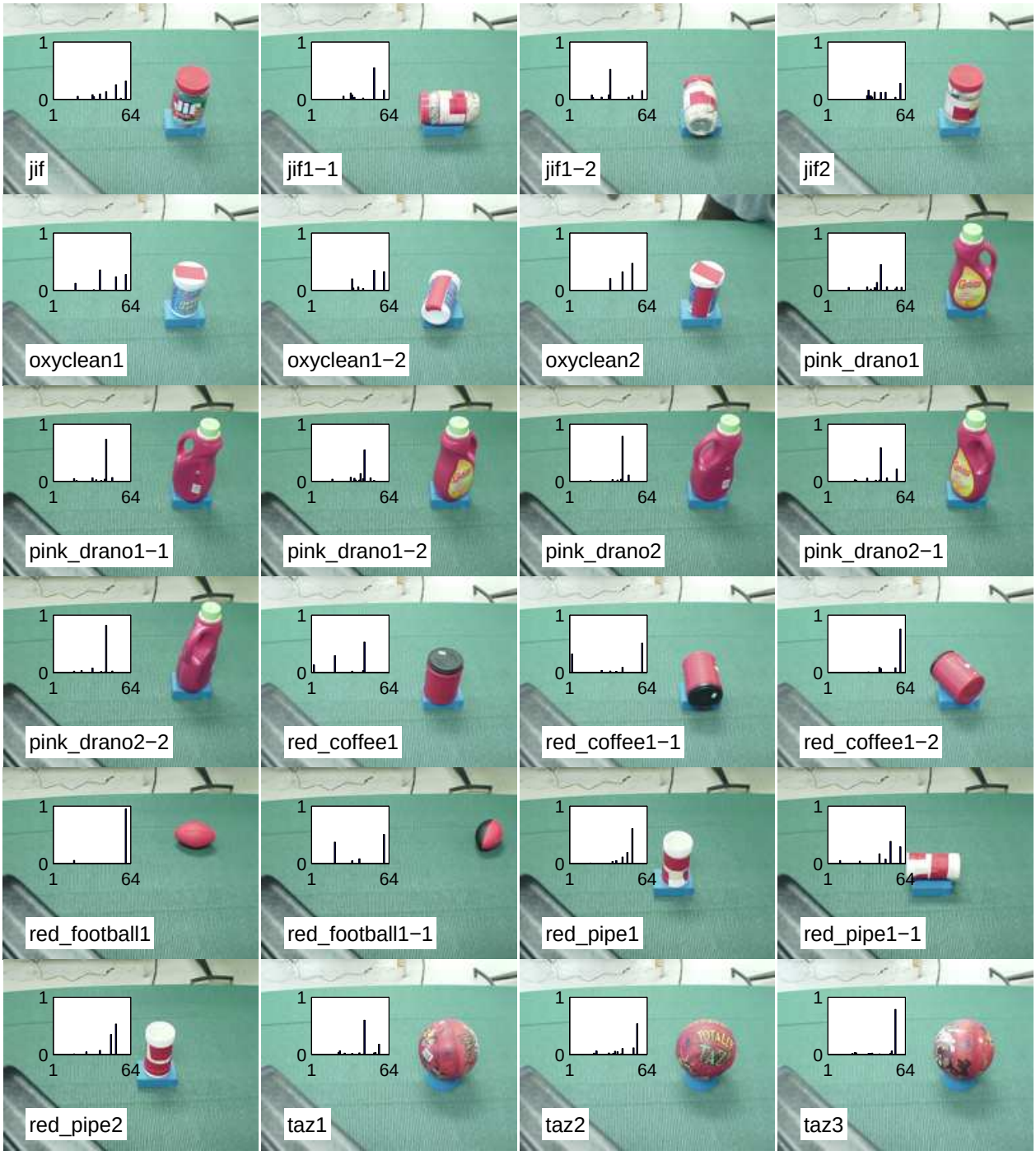

Figure B.1. There are 24 object presentations classified as Red.

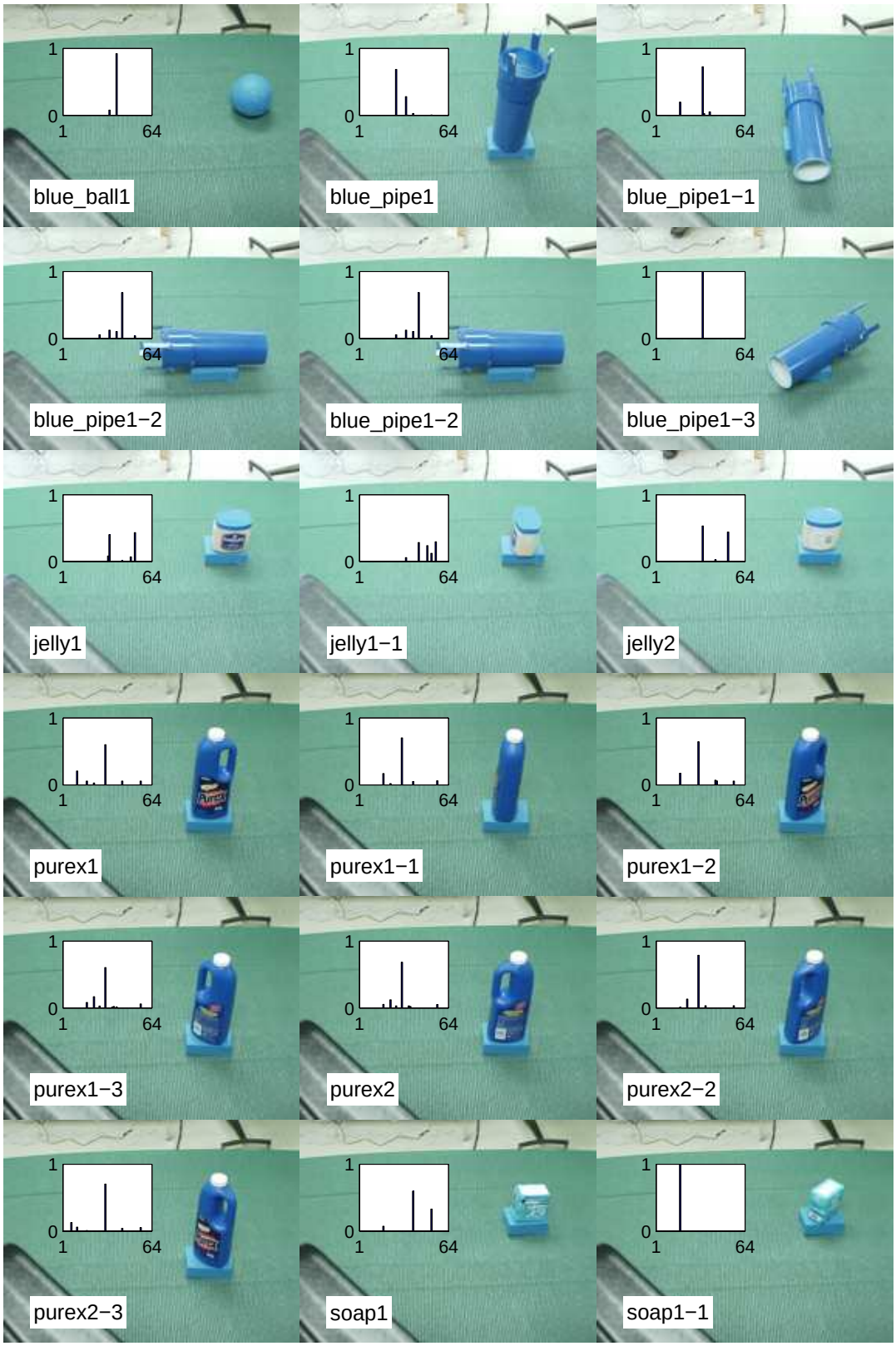

Figure B.2. There are 18 object presentations classified as Blue.

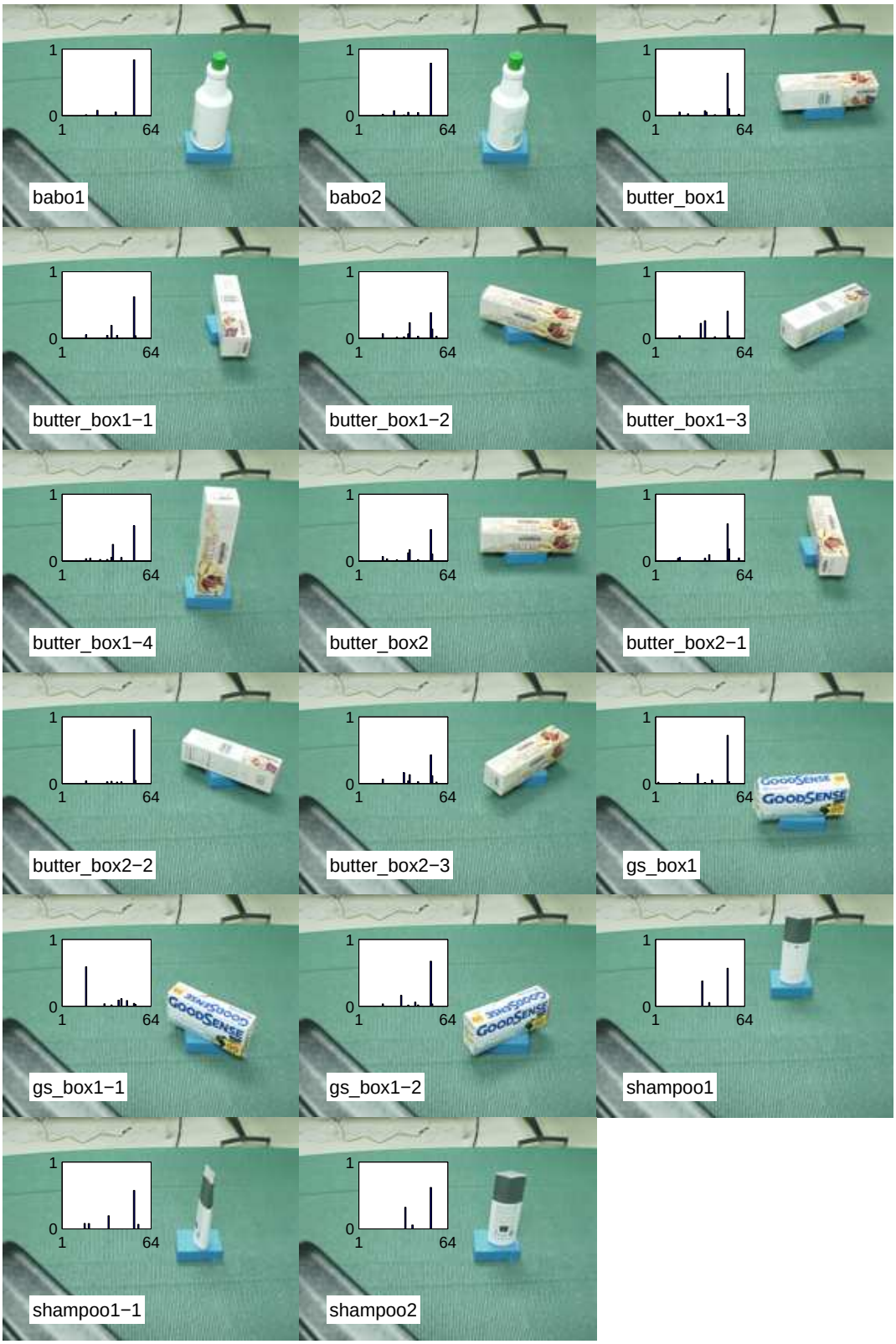

Figure B.3. There are 17 object presentations classified as White.

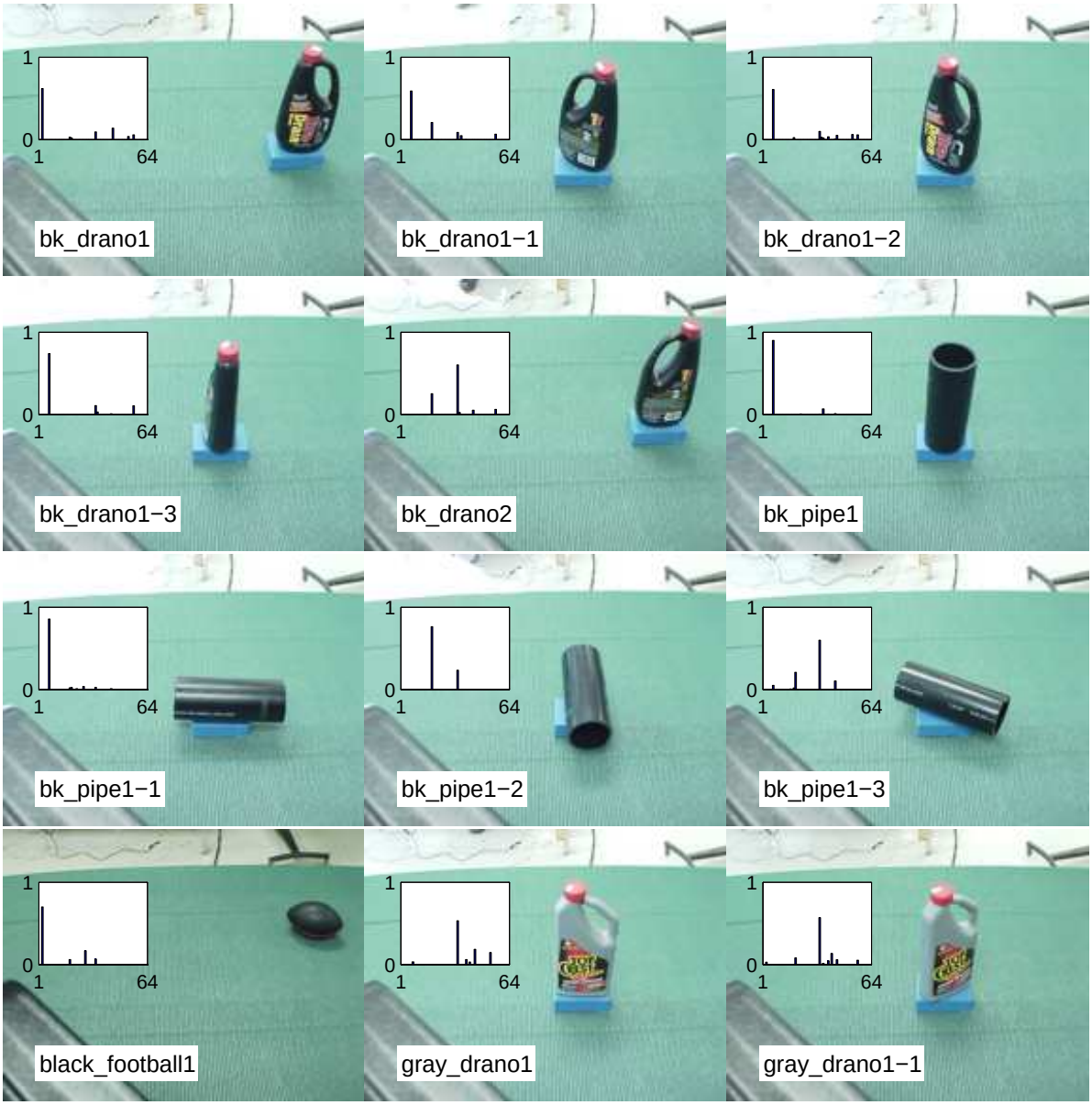

Figure B.4. There are 12 object presentations classified as Black.

Figure B.5. There are 38 object presentations classified as Small.

Figure B.6. There are 39 object presentations classified as Medium.

Figure B.7. There are 31 object presentations classified as Large.

BIBLIOGRAPHY

[1] Aleotti, Jacopo, and Caselli, Stefano. Trajectory clustering and stochastic approximation for robot programming by demonstration. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (August 2005), pp. 1029–1034.

[2] Amit, R., and Matarić, Maja. Learning movement sequences from demonstration. In *Proceedings of the International Conference on Development and Learning (ICDL)* (2002), pp. 203–208.

[3] Andrieu, Christophe, Freitas, Nando De, Doucet, Arnaud, and Jordan, Michael I. An introduction to MCMC for machine learning. *Machine Learning 50* (2003), 5–43.

[4] Angros, Jr., Richard Harrington. *Learning what to Instruct: Acquiring Knowledge from Demonstrations and Focussed Experimentation*. PhD thesis, University of Southern California, Los Angeles, CA, May 2000.

[5] Arbib, Michael A., Billard, Aude, Iacoboni, Marco, and Oztop, Erhan. Synthetic brain imaging: grasping, mirror neurons and imitation. *Neural Networks 13* (2000), 975–997.

[6] Asada, Haruhiko, and Izumi, Haruo. Direct teaching and automatic program generation for the hybrid control of robot manipulators. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* (1987), IEEE, pp. 1401–1406.

[7] Asada, Minoru, Yoshikawa, Yuichiro, and Hosoda, Koh. Learning by observation without three-dimensional reconstruction. In *Proceedings of Intelligent Autonomous Systems (IAS-6)* (2000), pp. 555–560.

[8] Astington, Janet Wilde. The paradox of intention: Assessing children's metarepresentational understanding. In *Intentions and Intentionality: Foundations of Social Cognition*, Bertram F. Malle, Louis J. Moses, and Dare A. Baldwin, Eds. The MIT Press, Cambridge, MA, 2001, ch. 4, pp. 85–103.

[9] Atkeson, Christopher G., and Schaal, Stefan. Learning tasks from a single demonstration. In *Proceedings of the International Conference on Robotics and Automation (ICRA)* (Albuquerque, NM, April 1997), vol. 3, IEEE, pp. 1706–1712.

[10] Atkeson, Christopher G., and Schaal, Stefan. Robot learning from demonstration. In *Proceedings of the Fourteenth International Conference on Machine Learning (ICML)* (1997), vol. 14, pp. 12–20.

[11] Baum, L.E. An inequality and associated maximization technique in statistical estimation for probabilistic functions of markov processes. *Inequalities 3*, 1 (1972), 1–8.

[12] Baum, LE, and Egon, JA. An inequality for rational functions with applications to somestatistical estimation problems. *Bull. Amer. Metereol. Soc 73* (1968), 211–227.

[13] Bentivegna, Darrin C., Ude, Aleš, Atkeson, Christopher G., and Cheng, Gordon. Humanoid robot learning and game playing using PC-based vision. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2002), vol. 3, pp. 2449–2454.

[14] Bernardo, José M., and Smith, Adrian F. M. *Bayesian Theory*. John Wiley & Sons, Ltd., 2000.

[15] Biernacki, Christophe, Celeux, Gilles, and Govaert, Gérard. Assessing a mixture model for clustering with the integrated completed likelihood. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22*, 7 (July 2000), 719–725.

[16] Billard, Aude, and Matarić, Maja J. Learning human arm movements by imitation: evaluation of a biologically inspired connectionist architecture. *Robotics and Autonomous Systems*, 941 (2001), 1–16.

[17] Bishop, Christopher M. Latent variable models. In *Learning in Graphical Models*, Michael I. Jordan, Ed. The MIT Press, 1998, pp. 371–403.

[18] Blackwell, David, and MacQueen, James B. Ferguson distributions via Poĺya urn schemes. *The Annals of Statistics 1*, 2 (March 1973), 353–355.

[19] Blei, David M., Ng, Andrew Y., and Jordan, Michael I. Latent Dirichlet allocation. *Journal of Machine Learning Research 3* (2003), 993–1022.

[20] Breazeal, Cynthis, and Scassellati, Brian. Robots that imitate humans. *Trends in Cognitive Sciences 6*, 11 (November 2002), 481–487.

[21] Buccino, G., Binkofskli, F., Fink, G. R., Fadiga, L., Fogassi, L., Gallese, V., Seitz, R. J., Zilles, K., Rizzolatti, G., and Freund, H.-J. Action observation activates premotor and parietal areas in a somatotopic manner: an fMRI study. *European Journal of Neuroscience 13* (2001), 400–404.

[22] Buccino, Giovanni, Vogt, Stefan, Ritzl, Afra, Fink, Gereon R., Zilles, Karl, Freund, Hans-Joachim, and Rizzolatti, Giacomo. Neural circuits underlying imitation learning of hand actions: An event-related fMRI study. *Neuron 42* (April 2004), 323–334.

[23] Bui, Hung H., Phung, Dinh Q., and Venkatesh, Svetha. Hierarchical hidden Markov models with general state hierarchy. In *Proceedings of the National Conference on Artificial Intelligence* (2004), no. 19, AAAI, pp. 324–329.

[24] Byrne, Richard W., and Russon, Anne E. Learning by imitation: A hierarchical approach. *Behavioral and Brain Sciences 21* (1998), 667–721.

[25] Canny, J.F. *The Complexity of Robot Motion Planning*. MIT Press, Cambridge, MA, 1988.

[26] Chen, Jason, and Zelinsky, Alex. Programing by demonstration: Coping with suboptimal teaching actions. *The International Journal of Robotics Research 22*, 5 (May 2003), 299–319.

[27] Chen, Jason R. Constructing task-level assembly strategies in robot programming by demonstration. *The International Journal of Robotics Research 24*, 12 (December 2005), 1073–1085.

[28] Coelho, J. *Multifingered Grasping: Haptic Reflexes and Control Context*. PhD thesis, University of Massachusetts, Amherst, MA, September 2001.

[29] Coelho, J., Araujo, EG, Huber, M, and Grupen, R. Dynamical categories and control policy selection. In *Proceedings of the ISIC/CIRA/ISAS'98 Conference* (Gaithersburg, MD, 1998), IEEE.

[30] Coelho, J., and Grupen, R. Online grasp synthesis. In *Proceedings of the Conference on Robotics and Automation* (Minneapolis, MN, April 1996), IEEE.

[31] Coelho, JA, and Grupen, RA. A control basis for learning multifingered grasps. *Journal of Robotic Systems 14*, 7 (1997), 545–557.

[32] Connolly, C., and Grupen, R. On the applications of harmonic functions to robotics. *Journal of Robotics Systems 10*, 7 (1993), 931–946.

[33] Craig, John J. *Introduction to Robotics: Mechanics and Control*, third ed. Series in Electrical and Computer Engineering: Control Engineering. Addison-Wesley, Reading, MA, 1989.

[34] Creem-Regehr, Sarah H., and Lee, James N. Neural representations of graspable objects: are tools special? *Cognitive Brain Research 22* (2005), 457–469.

[35] Csibra, Gergely. Teleological and referential understanding of action in infancy. *Philosophical Transactions of the Royal Society of London, Series B 358* (2003), 447–458.

[36] Csibra, Gergely, Bíró, Szilvia, Koós, Orsolya, and Gergely, György. One-year-old infants use teleological representations of actions productively. *Cognitive Science 27* (2003), 111–133.

[37] Csibra, Gergely, and Gergely, György. The teleological origins of mentalistic action explanations: A developmental hypothesis. *Developmental Science 1*, 2 (1998), 255–259.

[38] Dautenhahn, Kerstin, and Nehaniv, Chrystopher L., Eds. *Imitation in Animals and Artifacts*. The MIT Press, Cambridge, Massachusetts, 2002.

[39] de Granville, Charles, Southerland, Joshua, and Fagg, Andrew H. Learning grasp affordances through human demonstration. In *Proceedings of the International Conference on Development and Learning (ICDL)* (2006).

[40] Dean, Thomas, and Kanazawa, Keiji. A model for reasoning about persistance and causation. *Computational Intelligence 5*, 3 (1989), 142–150.

[41] Delson, Nathan, and West, Harry. Robot programming by human demonstration: Adaptation and inconsistency in constrained motion. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* (Minneapolis, Minnesota, April 1996), IEEE, pp. 30–36.

[42] Dempster, A. P., Laird, N. M., and Rubin, Donald B. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B (Methodology) 39*, 1 (1977), 1–38.

[43] Dillmann, Rüdiger. Teaching and learning of robot tasks via observation of human performance. *Robotics and Autonomous Systems 47*, 2,3 (June 2004), 109–116.

[44] Dillmann, Rüdiger, Kaiser, M., and Ude, Aleš. Acquisition of elementary robot skills from human demonstration. In *Proceedings of the International Symposium on Intelligent Robotic Systems (SIRS)* (Pisa, Italy, 1995).

[45] Dixon, Kevin R., Dolan, John M., and Khosla, Pradeep K. Predictive robot programming: Theoretical and experimental analysis. *The International Journal of Robotics Research 23*, 9 (September 2004), 955–973.

[46] Drumwright, Evan, Jenkins, Odest Chadwicke, and Matarić, Maja. Exemplar-based primitives for humanoid movement classification and control. In *Proceedings of the International Conference on Robotics and Automation (ICRA)* (New Orleans, LA, 2004), IEEE.

[47] Drumwright, Evan, and Matarić, Maja J. Generating and recognizing free-space movements in humanoid robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Las Vegas, NV, 2003), IEEE/RSJ.

[48] Duda, Richard O., Hart, Peter E., and Stork, David G. *Pattern Classification*, second ed. John Wiley & Sons, Inc., 2001.

[49] Ellis, Rob, and Tucker, Mike. Micro-affordance: The potentiation of components of action by seen objects. *British Journal of Psychology 91* (2000), 451–471.

[50] Escobar, Michael D., and West, Mike. Bayesian density estimation and inference using mixtures. *Journal of the American Statistical Association 90*, 430 (June 1995), 577–588.

[51] Fagg, Andrew H., and Arbib, Michael A. Modeling parietal-premotor interactions in primate control of grasping. *Neural Networks 11* (1998), 1277–1303.

[52] Ferguson, Thomas S. A Bayesian analysis of some nonparametric problems. *Annals of Statistics 1*, 2 (March 1973), 209–230.

[53] Fikes, R.E, and Nilsson, N.J. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence 2*, 3,4 (1971).

[54] Flanagan, J. Randall, and Johansson, Roland S. Actions plans used in action observation. *Nature 424*, 14 (August 2003), 769–771.

[55] Fogassi, Leonardo, Ferrari, Pier Francesco, Gesierich, Benno, Rozzi, Stefano, Chersi, Fabian, and Rizzolatti, Giacomo. Parietal lobe: From action organization to intention understanding. *Science 308* (April 2005), 662–667.

[56] Friedrich, H., Münch, S., Dillmann, R., Bocionek, S., and Sassin, M. Robot programming by demonstration (RPD): Supporting the induction by human interaction. *Machine Learning 23* (1996), 163–189.

[57] Friedrich, Holger, and Dillmann, Rüdiger. Robot programming based on a single demonstration and user intentions. In *Proceedings of the European Workshop on Learning Robots at ECML* (1995).

[58] Frith, Chris D., and Frith, Uta. Interacting minds — a biological basis. *Science 286* (November 1999), 1692–1695.

[59] Gallese, Vittorio, Keysers, Christian, and Rizzolatti, Giacomo. A unifying view of the basis of social cognition. *Trends in Cognitive Science 8*, 9 (September 2004), 396–403.

[60] Garland, Andrew, and Lesh, Neal. Learning hierarchical task models by demonstration. Tech. Rep. TR2003-01, Mitsubishi Research Laboratories, January 2003.

[61] Gelfand, Alan E., and Smith, Adrian F. M. Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association 85*, 410 (June 1990), 398–409.

[62] Gelman, Andrew, Carlin, John B., Stern, Hal S., and Rubin, Donald B. *Bayesian Data Analysis*, second ed. Texts in Statistical Science. Chapman & Hall/CRC, 2004.

[63] Gergely, György. What should a robot learn from an infant? Mechanisms of action interpretation and observational learning in infancy. In *Proceedings of the Third International Workshop on Epigenetic Robotics* (Boston, MA, August 2003), Christopher G. Prince, Luc Berthouze, Hideki Kozima, Daniel Bullock, Georgi Stojanov, and Christian Balkenius, Eds., Lund University Cognitive Studies, pp. 13–24.

[64] Gergely, György, Bekkering, Harold, and Király, Ildikó. Rational imitation in preverbal infants. *Nature 415* (February 2002), 755.

[65] Gergely, György, and Csibra, Gergely. Teleological reasoning in infancy: The infant's naive theory of rational action, a reply to Premack and Premack. *Cognition 63* (1997), 227–233.

[66] Gergely, György, and Csibra, Gergely. Teleological reasoning in infancy: the naïve theory of rational action. *Trends in Cognitive Science 7*, 7 (July 2003), 287–291.

[67] Gergely, György, Nádasdy, Zoltán, Csibra, Gergely, and Bíró, Szilvia. Taking the intentional stance at 12 months of age. *Cognition 56*, 2 (August 1995), 165–193.

[68] Ghahramani, Zoubin. Non-parametric Bayesian methods. Tutorial in Uncertainty in Artificial Intelligence (UAI), July 2005.

[69] Gibson, James J. The theory of affordances. In *Perceiving, Acting, and Knowing*, Robert Shaw and John Bransford, Eds. Lawrence Erlbaum Associates, Hillsdale, New Jersey, 1977, ch. 3, pp. 67–82.

[70] Griffiths, Thomas L., and Steyvers, Mark. Finding scientific topics. In *Proceedings of the National Academy of Sciences* (2004), vol. 101, pp. 5228–5235.

[71] Grudic, Gregory Z., and Lawrence, Peter D. Human-to-robot skill transfer using the SPORE approximation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* (Minneapolis, Minnesota, April 1996), vol. 4, IEEE, pp. 2962–2967.

[72] Grupen, R., and Souccar, K. Manipulability-based spatial isotropy: A kinematic reflex. In *Workshop on Mechatronical Computer Systems for Perception and Action* (Halmstad, SWEDEN, June 1-3 1993).

[73] Guyon, Isabelle, and Elisseeff, André. An introduction to variable and feature selection. *Journal of Machine Learning Research 3* (March 2003), 1157–1182.

[74] Hart, Stephen, and Grupen, Rod. Natural task decomposition with intrinsic potential fields. In *Proceedings of the International Conference on Intelligent Robotics and Systems (IROS)* (San Diego, CA, 2007).

[75] Hart, Stephen, Sen, Shiraj, and Grupen, Rod. Intrinsically motivated hierarchical manipulation. In *Proceedings of the International Conference on Robotics and Automation (ICRA)* (2008).

[76] Hart, Stephen W. *The Development of Hierarchical Knowledge in Robot Systems*. PhD thesis, University of Massachusetts Amherst, Sep 2009.

[77] Hart, Stephen W., Ou, Shichao, Sweeney, John D., and Grupen, Roderic A. A framework for learning declarative structure. In *Workshop on Manipulation for Human Environments, Robotics: Science and Systems* (Philadelphia, PA, August 2006).

[78] Hastings, W. K. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika 57*, 1 (April 1970), 97–109.

[79] Hayes, Gillian, and Demiris, John. A robot controller using learning by imitation. In *Proceedings of the International Symposium on Intelligent Robotic Systems (SIRS)* (Grenoble, France, July 1994).

[80] Heckerman, David. A tutorial on learning with Bayesian networks. In Jordan [90], pp. 301–354.

[81] Hovland, Geir E., Sikka, Pavan, and McCarragher, Brenan J. Skill acquisition from human demonstration using a hidden markov model. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* (Minneapolis, Minnesota, April 1996), IEEE, pp. 2706–2711.

[82] Huber M, Grupen RA. A feedback control structure for on-line learning tasks. *Journal of Robots and Autonomous Systems 22*, 3-4 (1997), 303–315.

[83] Iacoboni, Marco, Molnar-Szakacs, Istvan, Gallese, Vittorio, Buccino, Giovanni, Mazziotta, John C., and Rizzolatti, Giacomo. Grasping the intentions of others with one's own mirror neuron system. *PLoS Biology 3*, 3 (March 2005), e79.

[84] Iacoboni, Marco, Woods, Roger P., marcel Brass, Bekkering, Harold, Mazziotta, John C., and Rizzolatti, Giacomo. Cortical mechanisms of human imitation. *Science 286* (December 1999), 2526–2528.

[85] Ijspeert, Auke Jan, Nakanishi, Jun, and Schaal, Stefan. Learning rhythmic movements by demonstration using nonlinear oscillators. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Lausanne, Switzerland, September 2002), IEEE/RSJ, pp. 958–963.

[86] Ijspeert, Auke Jan, Nakanishi, Jun, and Schaal, Stefan. Movement imitation with nonlinear dynamical systems in humanoid robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* (Washington, D.C., May 2002), IEEE, pp. 1398–1403.

[87] Jenkins, Odest Chadwicke, and Matarić, Maja J. Deriving action and behavior primitives from human motion data. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Lausanne, Switzerland, September 2002), IEEE/RSJ, pp. 2251–2556.

[88] Jenkins, Odest Chadwicke, and Matarić, Maja J. A spatio-temporal extension to isomap nonlinear dimension reduction. In *Proceedings of the International Conference on Machine Learning (ICML)* (Banff, Canada, 2004), pp. 441–448.

[89] Jones, J.L., and Lozano-Pérez, T. Planning two-fingered grasps for pick-and-place operations on polyhedra. In *Proceedings of 1990 Conference on Robotics and Automation* (May 1990), IEEE, pp. 683–688.

[90] Jordan, Michael I., Ed. *Learning in Graphical Models*. Adaptive Computation and Machine Learning. The MIT Press, Cambridge, MA, 1998.

[91] Jordan, Michael I. Graphical models. *Statistical Science 19*, 1 (2004), 140–155.

[92] Jordan, Michael I. Dirichlet processes, Chinese restaurant processes and all that. Tutorial in Neural Information Processing Systems (NIPS), 2005.

[93] Juang, BH, and Rabiner, LR. Hidden markov models for speech recognition. *Technometrics 33*, 3 (1991), 251–272.

[94] Kaiser, M., and Dillmann, R. Building elementary robot skills from human demonstration. In *Proceedings of the International Conference on Robotics and Automation (ICRA)* (Minneapolis, Minnesota, April 1996), vol. 3, IEEE, pp. 2700–2705.

[95] Kang, Sing Bing, and Ikeuchi, Katsushi. Toward automatic robot instruction from perception–recognizing a grasp from observation. *IEEE Transactions on Robotics and Automation 9*, 4 (Aug 1993), 432–443.

[96] Kang, Sing Bing, and Ikeuchi, Katsushi. Toward automatic robot instruction from perception–temporal segmentation of tasks from human hand motion. *IEEE Transactions on Robotics and Automation 11*, 5 (Oct 1995), 670–681.

[97] Khatib, Oussama. Real-time obstacle avoidance for manipulators and mobile robots. In *Proceedings of the International Conference on Robotics and Automation (ICRA)* (Mar 1985), vol. 2, IEEE, pp. 500–505.

[98] Khatib, Oussama. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal of Robotics and Automation RA-3*, 1 (Feb 1987), 43–53.

[99] Koditschek, D.E., and Rimon, E. Robot navigation functions on manifolds with boundary. *Advances in Applied Mathematics 11*, 4 (1990), 412–442.

[100] Kohler, Evelyne, Keysers, Christian, Umiltá, M. Alessandra, Fogassi, Leonard, Gallese, Vittorio, and Rizzolatti, Giacomo. Hearing sounds, understanding actions: Action representation in mirror neurons. *Science 297* (August 2002), 846–848.

[101] Kuniyoshi, Yasuo, Inaba, Masayuki, and Inoue, Hirochika. Learning by watching: Extracting resuable task knowledge from visual observation of human performance. *IEEE Transactions on Robotics and Automation 10*, 6 (December 1994), 799–822.

[102] Kuniyoshi, Yasuo, Inoue, Hirochika, and Inaba, Masayuki. Design and implementation of a system that generates assembly programs from visual recognition of human action sequences. In *Proceedings of the International Workshop on Intelligent Robots and Systems (IROS)* (Ibaraki, Japan, Jul 1990), vol. 2, IEEE, pp. 567–574.

[103] Liapunov, A.M. *Stability of Motion.* Academic Press, New York, NY, 1966.

[104] Lopes, Manuel Cabido, and Santos-Victor, José. Visual learning by imitation with motor representations. *IEEE Transactions on Systems, Man, and Cybernetics, Part B 35*, 3 (June 2005), 438–449.

[105] Lowe, David G. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision 60*, 2 (2004), 91–110.

[106] Lozano-Pérez, Tomás, Jones, Joseph L., Mazer, Emmanuel, and O'Donnell, Patrick A. Task-level planning of pick-and-place robot motions. *Computer 22*, 3 (March 1989), 21–29.

[107] MacKay, David J. C. *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press, 1995.

[108] Mardia, Kanti V., and Jupp, Peter E. *Directional Statistics*, second ed. John Wiley & Sons, Ltd., 2000.

[109] Matarić, Maja J. Getting humanoids to move and imitate. *IEEE Intelligent Systems* (July/August 2000), 18–24.

[110] Matarić, Maja J. Sensory-motor primitives as a basis for imitation: Linking perception to action and biology to robotics. In Dautenhahn and Nehaniv [38], ch. 15, pp. 392–422.

[111] McLachlan, G. J. On bootstrapping the likelihood ratio test statistics for the number of components in a normal mixture. *Applied Statistics 36*, 3 (1987), 318–324.

[112] Meltzoff, Andrew N. Infant imitation after a 1-week delay: Long-term memory for novel acts and multiple stimuli. *Developmental Psychology 24*, 4 (1988), 470–476.

[113] Metropolis, Nicholas, Rosenbluth, Arianna W., Rosenbluth, Marshall N., Teller, Augusta H., and Teller, Edward. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics 21*, 6 (June 1953), 1087–1092.

[114] Metropolis, Nicholas, and Ulam, S. The Monte Carlo method. *Journal of the American Statistical Association 44*, 247 (September 1949), 335–341.

[115] Miyata, Natsuki, Oguri, Kenichiro, Ota, Jun, and Arai, Tamio. Human lift-up motion generation based on identification of time-invariant performance index. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (Lausanne, Switzerland, September 2002), IEEE/RSJ, pp. 2503–2508.

[116] Müller, Peter, and Quintana, Fernando A. Nonparametric bayesian data analysis. *Statistical Science 19*, 1 (2004), 95–110.

[117] Mussa-Ivaldi, F. A., Giszter, S. F., and Bizzi, E. Linear combinations of primitives in vertebrate motor control. *Proceedings of the National Academy of Science USA 91* (Aug 1994), 7534–7538.

[118] Nakamura, Yoshihiko. *Advanced Robotics: Redundancy and Optimization*. Addison-Wesley Longman Publishing Co., Inc., 1990.

[119] Nakamura, Yoshihiko. *Advanced Robotics: Redundancy and Optimization*. Addison-Wesley Publishing Company, Inc., 1991.

[120] Nakanishi, Jun, Cory, Rick, Mistry, Michael, Peters, Jan, and Schaal, Stefan. Operational space control: A theoretical and empirical comparison. *The International Journal of Robotics Research 27*, 6 (June 2008), 737–757.

[121] Neal, Radford M. Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics 9*, 2 (2000), 249–265.

[122] Nehaniv, Chrystopher L., and Dautenhahn, Kerstin. The correspondence problem. In Dautenhahn and Nehaniv [38], ch. 2.

[123] Ng, Andrew Y., and Jordan, Michael I. On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. In *Advances in Neural Information Processing Systems (NIPS)* (2001), vol. 2, MIT Press, pp. 841–848.

[124] Nicolescu, Monica N., and Matarić, Maja J. Learning and interacting in human-robot domains. *Special Issue of IEEE Transactions on Systems, Man, and Cybernetics, Part A: Systems and Humans 31*, 5 (September 2001), 419–430.

[125] Pardowitz, Michael, Zöllner, Raoul, and Dillmann, Rüdiger. Learning sequential constraints of tasks from user demonstrations. In *Proceedings of the IEEE-RAS International Conference on Humanoid Robots* (2005), pp. 424–429.

[126] Philipose, Matthai, Fishkin, Kenneth P., Perkowitz, Mike, Patterson, Donald J., Fox, Deiter, Kautz, Henry, and Hähnel, Dirk. Inferring activities from interactions with objects. *Pervasive Computing 3*, 4 (Oct–Dec 2004), 50–57.

[127] Platt, Rob. *Learning and Generalizing Control Based Grasping and Manipulation Skills*. PhD thesis, University of Massachusetts Amherst, Amherst, MA, September 2006.

[128] Platt, Rob, Fagg, Andrew H., and Grupen, Rod. Nullspace composition of control laws for grasping. In *Proceedings of the International Conference on Intelligent Robotics and Systems (IROS)* (2002).

[129] Pook, Polly K., and Ballard, Dana H. Recognizing teleoperated manipulations. In *Proceedings of the International Conference on Robotics and Automation (ICRA)* (1993), pp. 578–585.

[130] Povinelli, Daniel J., and Preuss, Todd M. Theory of mind: evolutionary history of a cognitive specialization. *Trends in Neuroscience 18*, 9 (1995), 418–424.

[131] Rabiner, L.R., and Juang, B.H. Introduction to hidden markov models. *IEEE ASSP MAG. 3*, 1 (1986), 4–16.

[132] Rabiner, L.R., and Juang, B.H. *Fundamentals of speech recognition.* Prentice Hall, 1993.

[133] Rimon, E., and Koditschek, D. The construction of analytic diffeomorphisms for exact robot navigation on star worlds. In *Proceedings of the 1989 Conference on Robotics and Automation* (Scottsdale, AZ, May 1989), vol. 1, IEEE, pp. 21–26.

[134] Rimon, Elon, and Koditschek, Daniel E. Exact robot navigation using artificial potential functions. *IEEE Transactions of Robotics and Automation 8*, 5 (Oct 1992), 501–518.

[135] Rizzolatti, Giacomo, Fadiga, Luciano, Gallese, Vittorio, and Fogassi, Leonardo. Premotor cortex and the recognition of motor actions. *Cognitive Brain Research 3* (1996), 131–141.

[136] Rosen-Zvi, Michal, Griffiths, Thomas, Steyvers, Mark, and Smyth, Padhraic. The author-topic model for authors and documents. In *Proceedings of the* 20^{th} *Conference on Uncertainty in Artificial Intelligence (UAI)* (Banff, Canada, 2004), AUAI Press, pp. 487–494.

[137] Roweis, Sam, and Ghahramani, Zoubin. A unifying review of linear gaussian models. *Neural Computation 11* (1999), 305–345.

[138] Rybski, Paul E., and Voyles, Richard M. Interactive task training of a mobile robot through human gesture recognition. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)* (Detroit, MI, May 1999), vol. 1, pp. 664–669.

[139] Saxena, Ashutosh, Driemeyer, Justin, Kearns, Justin, Osondu, Chioma, and Ng, Andrew Y. Learning to grasp novel objects using vision. In *Proceedings of the* 10^{th} *International Symposium on Experimental Robotics (ISER)* (Rio de Janeiro, Brazil, July 2006).

[140] Schaal, Stefan. Is imitation learning the route to humanoid robots? *Trends in Cognitive Science 3*, 6 (1999), 233–242.

[141] Schaal, Stefan, Ijspeert, Auke, and Billard, Aude. Computational approaches to motor learning by imitation. *Philosophical Transaction of the Royal Society of London: Series B, Biological Sciences* (2003), 537–547.

[142] Schaal, Stefan, Peters, J., Nakanishi, Jun, and Ijspeert, A. Learning movement primitives. In *International Symposium on Robotics Research (ISRR2003)* (Ciena, Italy, 2004), Springer Tracts in Advanced Robotics, Springer.

[143] Sethuraman, Jayaram. A constructive definition of Dirichlet priors. *Statistica Sinica 4*, 2 (1994), 639–650.

[144] Sivic, Josef, Russell, Bryan C., Efros, Alexei A., Zisserman, Andrew, and Freeman, William T. Discovering objects and their location in images. In *Proceedings of the* 10^{th} *International Conference on Computer Vision (ICCV)* (2005), IEEE.

[145] Stoytchev, Alexander. Behavior-grounded representation of tool affordances. In *Proceedings of the International Conference on Robotics and Automation (ICRA)* (2005), IEEE.

[146] Sudderth, Erik B. *Graphical Models for Visual Object Recognition and Tracking.* PhD thesis, Masschusetts Institute of Technology, May 2006.

[147] Sudderth, Erik B., Torralba, Antonio, Freeman, William T., and Willsky, Alan S. Describing visual scences using transformed Dirichlet processes. In *Proceedings of Neural Information Processing Systems (NIPS)* (2005).

[148] Sudderth, Erik B., Torralba, Antonio, Freeman, William T., and Willsky, Alan S. Learning hierarchical models of scenes, objects, and parts. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)* (October 2005), vol. 2, IEEE, pp. 1331–1338.

[149] Teh, Yee Whye, Jordan, Michael I., Beal, Matthew J., and Blei, David M. Hierarchical Dirichlet processes. *Journal of the American Statistical Association 101*, 476 (2006), 1566–1581.

[150] Tominaga, Hirohisa, and Ikeuchi, Katsushi. Acquiring manipulation skills through observation. In *Proceedings of the International Conference on Multisensor Fusion and Integration for Intelligent Systems* (Taipei, Taiwan, Aug 1999), IEEE.

[151] Ude, Aleš, Atkeson, Christopher G., and Riley, Marcia. Programming full-body movements for humanoid robots by observation. *Robotics and Autonomous Systems 47*, 2–3 (June 2004), 93–108.

[152] Uppala, S., Karuppiah, D., Brewer, M., Ravela, S., and Grupen, R. A. On viewpoint control. In *submitted to the IEEE Conference on Robotics and Automation* (Washington, DC, May 2002), IEEE.

[153] van Lent, Michael, and Laird, John E. Learning procedural knowledge through observation. In *Proceedings of the International Conference on Knowledge Capture (K-CAP)* (Victoria, Canada, 2001), pp. 179–186.

[154] Vertut, Jean, and Coiffet, Philippe. *Teleoperation and Robotics: Evolution and Development*, vol. 3a of *Robot Technology*. Prentice-Hall, Englewood Cliffs, NJ, 1984.

[155] Visalberghi, Elisabetta, and Fragaszy, Dorothy. "do monkeys ape?" – ten years after. In Dautenhahn and Nehaniv [38], ch. 18, pp. 471–499.

[156] Voyles, R.M., and Khosla, P.K. A multi-agent system for programming robots by human demonstration. *Integrated Computer-Aided Engineering 8*, 1 (2001), 59–67.

[157] Voyles, R.M., Morrow, J.D., and Khosla, P.K. Gesture-based programming for robotics: Human augmented software adaptation. *IEEE Intelligent Systems 14*, 6 (1999), 22–29.

[158] Yang, Jie, Xu, Yangsheng, and Chen, C.S. Hidden Markov model approach to skill learning and its application to telerobotics. *IEEE Transactions on Robotics and Automation 10*, 5 (Oct 1994), 621–631.

[159] Yeasin, Mohammed, and Chaudhuri, Subhasis. Toward automatic robot programming: Learning human skill from visual data. *IEEE Transactions on Systems, Man, and Cybernetics–Part B: Cybernetics 30*, 1 (February 2000), 180–185.

[160] Yokokohji, Yasuyoshi, Kitaoka, Yuki, and Yoshikawa, Tsuneo. Motion capture from demonstrator's viewpoint and its application to robot teaching. *Journal of Robotic Systems 22*, 2 (February 2005), 87–97.

[161] Yoshikawa, T. Manipulability of robotic mechanisms. *Journal of Robotics Research 4*, 2 (Summer 1985), 3–9.